

USER GUIDE

Essential Studio

for EJ2 ASP.NET MVC

Version - v24.1.41 | Release Date - December 18, 2023

Syncfusion ASP.NET MVC UI (Essential JS 2)	92
How to best read this user guide	92
Control List	92
table	92
table	93
Getting Help	95
See also	95
System Requirements for ASP.NET MVC EJ2 Components	96
Integrated Development Environment (IDE)	96
Frameworks	96
Browser Compatibility in ASP.NET MVC	96
See also	96
NuGet Packages for ASP.NET MVC UI controls	97
Available NuGet packages	97
Syncfusion.EJ2.MVC5	97
Syncfusion.EJ2.WordEditor.AspNet.Mvc5	97
Syncfusion.EJ2.PdfViewer.AspNet.Mvc5	97
Syncfusion.EJ2.SpellChecker.AspNet.Mvc5	98
Syncfusion.EJ2.Spreadsheet.AspNet.MVC5	98
Syncfusion.EJ2.GridExport.MVC5	98
Getting Started	98
Getting Started with ASP.NET MVC using HTML Helper	98
Prerequisites	98
Create ASP.NET MVC application with HTML helper	98
Install ASP.NET MVC package in the application	98
Add namespace	99
Add stylesheet and script resources	99
Add ASP.NET MVC Calendar control	100
Getting Started with ASP.NET Core MVC using HTML Helper	100
Prerequisites	100
Create ASP.NET Core MVC application	100
Install ASP.NET Core package in the application	100
Add Syncfusion ASP.NET Core namespace	101
Add stylesheet and script resources	101
Add ASP.NET Core Calendar control	102

See also	102
Getting Started with ASP.NET Core MVC using Tag Helper	102
Prerequisites	102
Create ASP.NET Core MVC application	102
Install ASP.NET Core package in the application	102
Add Syncfusion ASP.NET Core Tag Helper	103
Add stylesheet and script resources	103
Add ASP.NET Core Calendar control	104
See also	104
Visual Studio extensions	104
Create project	104
Installation	109
Web Installer	109
Downloading Syncfusion ASP.NET MVC EJ2 web installer.....	109
Installing Syncfusion ASP.NET MVC EJ2 Web Installer.....	112
Offline Installer	125
Downloading Syncfusion ASP.NET MVC EJ2 offline installer	125
Installing Syncfusion ASP.NET MVC EJ2 Offline Installer	128
Install Syncfusion ASP.NET MVC JS2 NuGet packages	137
Overview	137
Installation using Package Manager UI	138
Installation using Package Manager Console.....	139
Common Installation Errors	141
Unlocking the license installer using the trial key.....	141
License has expired	142
Unable to find a valid license or trial	142
Unable to install because of another installation	143
Unable to install due to controlled folder access	144
Licensing.....	145
Syncfusion Licensing Overview	145
Difference between unlock key and license key.....	145
Registering Syncfusion license keys in Build server	145
See Also	146
Generate Syncfusion ASP .NET MVC EJ2 License key	146
Claim License key	146

See Also	148
Register Syncfusion License key in ASP .NET MVC EJ2 application.....	148
ASP . NET MVC	148
Syncfusion Licensing Errors.....	148
Licensing errors	148
Licensing errors from version 16.2.0* to 20.3.0*	150
Could not load Syncfusion.Licensing.dll assembly version	151
Syncfusion Licensing FAQ.....	151
Is an internet connection required for Syncfusion Essential Studio license validation?	151
How to upgrade from Trial version after purchasing a license.....	151
Where can I get a License key.....	152
Registering Syncfusion account for direct NuGet.org user	152
Registering license key for Syncfusion JavaScript Components used in ASP.NET MVC App	152
Upgrade.....	153
Upgrading Syncfusion ASP .NET MVC EJ2 installer to a latest version	153
Upgrading to the latest version	153
Upgrading from trial version to license version.....	153
Upgrading Syncfusion ASP.NET MVC JS2 components to latest version	154
Upgrading Syncfusion ASP.NET MVC NuGet packages to a latest version	155
Upgrade NuGet packages through Package Manager UI	155
Upgrade NuGet packages through Package Manager Console	157
Upgrading Syncfusion ASP.NET MVC Extensions	158
Upgrading to the latest version	158
Appearance	159
Syncfusion ASP.NET MVC Themes	159
Reference themes in ASP.NET MVC application	160
CDN Reference	160
Change theme dynamically	161
Size Mode for ASP.NET MVC Controls	162
Size mode for application	163
Size mode for a Control.....	163
Change size mode for application at runtime.....	163
Change size mode for a control at runtime	165
See Also	167
Overview in ASP.NET MVC Controls	168

Customizing theme color from theme studio.....	168
Import previously changed settings into the theme studio	173
Common Variables	174
Icons Library	188
Steps to Use Icon.....	188
Icons Size	189
Tooltip for icons	190
Icon appearance customization	190
Third party icons integration.....	191
HTML attribute support	191
Using icons directly in HTML element.....	192
Icons list	192
Common.....	193
Reference Scripts in ASP.NET MVC Application.....	193
CDN Reference	193
Node Package Manager (NPM).....	194
Custom Resource Generator.....	202
Custom Resource Generator for Syncfusion ASP.NET MVC	202
Download the selected control resources	207
How to use custom resources in the ASP.NET MVC application	211
Import previously generated settings into the CRG	211
Strongly-Typed HTML Helper	213
Server Side Validation	216
Accessibility support with ADA compliance in ASP.NET MVC controls	217
Section 508.....	217
Keyboard navigation	217
Localization (Multi-Language) support in ASP.NET MVC	218
Localization of Syncfusion ASP.NET MVC Controls	219
Statically set the culture	219
Dynamically set the culture	220
Changing current locale	222
See also	223
Globalization in ASP.NET MVC Control	223
Loading Culture Data	223
Changing Global Culture and Currency Code.....	227

Manipulating Numbers	227
Manipulating DateTime	231
Right-To-Left support in Syncfusion ASP.NET MVC controls	235
Enable RTL for all controls.....	236
Enable RTL to individual control	236
State Persistence in ASP.NET MVC	237
State Persistence Supported controls and Properties	237
Animation in Syncfusion ASP.NET MVC	240
Animation Supported controls.....	240
Animating a HTML Element	241
Enable or disable animation globally	242
Drag and Drop support in Syncfusion ASP.NET MVC controls.....	243
Drag and Drop Supported controls	243
Initializing custom Draggable element	243
Creating Droppable zone	244
Defining Drop Action.....	245
See also	246
Compatibility with Syncfusion ASP.NET MVC (Essential JS 1).....	246
Adding application	246
Style compatibility.....	246
Script compatibility	247
Adding compatibility	247
Adding script manager	247
Declare Essential JS 1 and Essential JS 2 controls	249
Template Engine	249
Compiling	249
Available Template Syntax.....	250
Custom Helper	250
Deployment in ASP.NET MVC	251
Publish ASP.NET MVC Application with Visual Studio	251
Publish ASP.NET MVC Application with CLI	254
See also	254
Input Form Validation	254
How to Validate Syncfusion ASP.NET MVC UI Controls.....	254
ASP.NET MVC Form Validation Supported Controls	257

See Also	257
How To	257
How to troubleshoot compile-time and run-time errors	257
Visual Studio Integration.....	258
Visual Studio Integration.....	258
Overview	258
IMPORTANT	258
Download and Installation	260
Prerequisites	260
Install through the Visual Studio Manage Extensions	260
Install from the Visual Studio Marketplace	261
Visual Studio extensions	263
Create project	263
Converting ASP.NET MVC application to ASP.NET MVC application	268
Upgrading Syncfusion ASP.NET MVC application to latest version	271
IMPORTANT	271
Create Sample in ASP.NET MVC Application	275
Project Configuration	279
Scaffolding in ASP.NET MVC Application	280
Check for Updates in ASP.NET MVC Application	286
Syncfusion Notifications	288
Notification Configuration	288
Notification Types	289
Accordion	290
Getting Started with ASP.NET MVC Accordion Control	290
Prerequisites	291
Create ASP.NET MVC application with HTML helper	291
Install ASP.NET MVC package in the application	291
Add namespace.....	291
Add stylesheet and script resources	291
Register Syncfusion script manager	292
Add ASP.NET MVC Accordion control	292
Render the Accordion using content template.....	293
Expand Mode in ASP.NET MVC Accordion Control.....	296
Single	296

Multiple	297
See also	298
Accessibility in ASP.NET MVC Accordion Control	298
ARIA attributes.....	299
Keyboard interaction	300
Ensuring accessibility	300
See also	300
CSS Structure in Accordion Control	300
Customizing Accordion	300
Customizing the list items	301
Customizing Accordion's header.....	301
Customizing Accordion's expand and collapse icons.....	301
Customizing the hover state of Accordion control	301
Customizing selected item of Accordion control	302
How To	302
Set the nested Accordion.....	302
Load content through Ajax	304
To keep single pane open always	305
Create wizard using Accordion	306
Load accordion with DataSource	312
Customize Accordion expand or collapse animation behavior.....	313
Load the content as partial view to Accordion	315
Migration from Essential JS 1.....	317
Accessibility	317
AjaxSettings.....	317
Animation.....	318
Items	319
Common.....	321
Accumulation Chart	323
Getting Started with ASP.NET MVC AccumulationChart Control	323
Prerequisites	323
Create ASP.NET MVC application with HTML helper.....	323
Install ASP.NET MVC package in the application	323
Add namespace.....	323
Add stylesheet and script resources	323

Register Syncfusion script manager	324
Add ASP.NET MVC AccumulationChart control	324
Pie Chart.....	326
Radius Customization.....	326
Pie Center.....	327
Various Radius Pie Chart	328
Doughnut Chart.....	329
Start and End angles	330
Color & Text Mapping	331
Customization	332
Hide pie or doughnut border	333
Color Palette	333
Multi-level pie chart.....	334
See Also	339
Pyramid Chart	339
Mode.....	340
Size	341
Gap Between the Segments.....	342
Explode.....	343
Customization	343
See Also	344
Funnel Chart.....	344
Size	345
Neck Size	346
Gap Between the Segments.....	347
Explode.....	347
Smart Data Label.....	348
Customization	350
See Also	351
Data Label in ASP.NET MVC Accumulation Chart Component	351
Positioning.....	352
Smart labels.....	352
Data Label Template	353
Connector Line	354
Text Mapping	355

Format.....	356
Customization	357
Text wrap	358
Show percentages in data labels of pie chart	359
Grouping	361
Pie Grouping.....	362
Customization	365
Empty Points	367
Customization	368
Annotation	369
Region	370
Co-ordinate Units	371
Alignment.....	371
Tooltip in ASP.NET MVC Accumulation chart component.....	372
Header.....	373
Format.....	374
Tooltip format.....	375
Fixed tooltip	376
Customization	376
To customize individual tooltip.....	377
Tooltip mapping name	378
Legend in ASP.NET MVC Accumulation Chart Component.....	379
Position and Alignment.....	380
Legend Reverse	381
Legend Shape	382
Legend Size.....	383
Legend Item Size	384
Paging for Legend.....	384
Legend Text Wrap	385
Enable Animation	386
Legend Title.....	387
Arrow Page Navigation	388
Legend Item Padding	389
Center label in ASP.NET MVC Accumulation Chart Component.....	390
Center label.....	390

Hover text	391
Customization	391
Title	392
Title Customization	393
SubTitle	394
SubTitle Customization	395
Print and Export	396
Print.....	396
Export.....	397
Accessibility in ASP.NET MVC Accumulation chart component.....	398
WAI-ARIA attributes.....	399
Keyboard interaction	400
Ensuring accessibility	400
See also	400
Migration from Essential JS 1.....	400
Accumulation Chart	400
Annotation	402
Series.....	403
DataLabel	407
Legend.....	409
Methods.....	412
Events.....	412
AppBar	418
Getting Started with ASP.NET MVC AppBar Control	418
Prerequisites	418
Create ASP.NET MVC application with HTML helper.....	418
Install ASP.NET MVC package in the application	419
Add namespace.....	419
Add stylesheet and script resources	419
Register Syncfusion script manager	419
Add ASP.NET MVC AppBar control	420
Size and Color with ASP.NET MVC AppBar Control	420
Size	420
Color.....	423
Accessibility in ASP.NET MVC AppBar Control.....	426

Keyboard interaction	427
Ensuring accessibility	427
See also	427
Positioning in ASP.NET MVC AppBar Control	427
Design User Interface with ASP.NET MVC AppBar Control	432
Spacer.....	432
Separator.....	432
Media Query	433
Designing AppBar with Menu	435
Designing AppBar with Buttons	436
Designing AppBar with SideBar.....	437
Styles and Appearances in ASP.NET MVC AppBar Control	441
CssClass	441
HtmlAttributes	442
Auto Complete	443
Getting Started with ASP.NET MVC AutoComplete Control	443
Prerequisites	443
Create ASP.NET MVC application with HTML helper.....	443
Install ASP.NET MVC package in the application	443
Add namespace.....	443
Add stylesheet and script resources	443
Register Syncfusion script manager	444
Add ASP.NET MVC AutoComplete control.....	444
Binding data source	445
Custom values.....	445
Configure the suggestion list	445
See also	446
Data binding in ASP.NET MVC AutoComplete Control	446
Bind to local data	446
Bind to remote data.....	448
See Also	454
Templates in ASP.NET MVC AutoComplete Control	454
Item template	454
Group template.....	455
Header template	455

Footer template	457
No records template	457
Action failure template	458
See also	458
Virtualization in AutoComplete Component	459
Binding local data	459
Binding remote data	460
Grouping	460
Grouping in ASP.NET MVC AutoComplete Control.....	461
Customization	462
Filtering in ASP.NET MVC AutoComplete Control.....	462
Change the filter type	462
Filter item count.....	463
Limit the minimum filter character.....	463
Case sensitive filtering	464
Diacritics filtering	465
See also	465
Localization in ASP.NET MVC AutoComplete Control.....	465
Loading translations.....	465
See also	466
CSS structures in AutoComplete Control	466
Customizing the appearance of wrapper element	466
Customizing the dropdown icon's color	467
Customizing the focus color	467
Customizing the outline theme's focus color	467
Customizing the disabled component's text color	468
Customizing the float label element's focusing color	468
Customizing the color of the placeholder text	468
Customizing the placeholder to add mandatory indicator(*).....	468
Customizing the text selection color.....	469
Customizing the background color of focus, hover, and active item's	469
Customizing the appearance of pop-up element	469
Accessibility in ASP.NET MVC AutoComplete Control	469
WAI-ARIA attributes.....	470
Keyboard interaction	471

Ensuring accessibility	472
See also	472
How To	473
Autofill supported with AutoComplete	473
Show the list items with icons	473
Custom highlight search.....	477
Filter using both text and value field	478
AutoCompleteFor.....	478
Migration from Essential JS 1.....	481
DataBinding.....	481
Filtering	482
Placeholder	482
Popup.....	482
CSS.....	484
Grouping	484
Localization	484
Template	484
Sorting.....	485
Accessibility.....	485
Selection.....	485
Miscellaneous	486
Common.....	486
Avatar	488
Getting Started with ASP.NET MVC Avatar Control.....	488
Prerequisites	488
Create ASP.NET web application with HTML helper.....	488
Add stylesheet.....	488
Add ASP.NET MVC Avatar control.....	488
See also	489
Types and Styles in ASP.NET MVC Avatar Control	489
Avatar size	489
Avatar types	490
How To	492
Avatar Customization.....	492
Integrate avatar into ListView.....	499

Integrate avatar into Badge	502
Badge	506
Getting Started with ASP.NET MVC Badge Control	506
Prerequisites	506
Create ASP.NET web application with HTML helper.....	507
Add stylesheet.....	507
Add ASP.NET MVC Badge control	507
See also	508
Types in ASP.NET MVC Badge Control	508
Badge styles	508
Badge types.....	511
How To	526
Customization in ASP.NET MVC Badge Control	526
Integrate Badge into ListView	532
Dynamic Badge Content.....	537
Barcode	541
Getting Started with ASP.NET MVC Barcode Control	541
Prerequisites	541
Create ASP.NET MVC application with HTML helper.....	541
Install ASP.NET MVC package in the application	541
Add namespace.....	542
Add stylesheet and script resources	542
Register Syncfusion script manager	542
Add ASP.NET MVC Barcode control	543
Adding QR Generator control	543
Adding Datamatrix Generator control	543
ASP.NET MVC BarcodeGenerator Control	544
Code39	544
Code39 Extended	545
Code 11	545
Codabar	546
Code 32	547
Code 93	547
Code 93 Extended	548
Code 128	548

Customizing the Barcode color	549
Customizing the Barcode dimension	550
Customizing the text	550
ASP.NET MVC QR Code generator Control	551
QR Code	551
Customizing the QR Code color	552
Customizing the QR Code dimension.....	552
Customizing the text	553
Data Matrix generator	554
Data Matrix	554
Customizing the DataMatrix color	554
Customizing the DataMatrix dimension	555
Customizing the text	555
Export.....	556
Export.....	556
Breadcrumb	557
Getting Started with ASP.NET MVC Breadcrumb Control	557
Prerequisites	557
Create ASP.NET MVC application with HTML helper.....	557
Install ASP.NET MVC package in the application	557
Add namespace.....	557
Add stylesheet and script resources	557
Register Syncfusion script manager	558
Add ASP.NET MVC Breadcrumb control	558
Add Items to the Breadcrumb Control	558
Enable or Disable Navigation	560
Data Binding in Breadcrumb	562
Items based on current Url	562
Customize text when generated items using Url.....	563
Icons	563
Loading icon in BreadcrumbItem.....	563
Icon Position.....	567
Icon Only	569
Show icon only for first item.....	570
Icons in Breadcrumb	572

Navigation	572
URL	572
Enable navigation for last Breadcrumb item	573
Open URL in new page or tab	574
Overflow Mode in breadcrumb	575
Overflow Mode	575
Collapsed	575
Menu	577
Wrap	578
Scroll	580
Hidden	581
None	583
Templates in Breadcrumb	583
Item Template	583
Separator Template	584
Customize Specific Item Template	585
Accessibility in ASP.NET MVC Breadcrumb control	587
WAI-ARIA attributes	588
Keyboard interaction	588
Ensuring accessibility	588
See also	589
Bullet Chart	589
Getting Started with ASP.NET MVC Bullet Chart Control	589
Prerequisites	589
Create ASP.NET MVC application with HTML helper	589
Install ASP.NET MVC package in the application	589
Add namespace	589
Add script resources	590
Register Syncfusion script manager	590
Add ASP.NET MVC Bullet Chart control	590
Bullet Chart With Data	590
Add Bullet Chart Title	591
Ranges	592
Tooltip	592
Bullet chart dimensions	593

Size for container	593
Size for bullet chart	594
Axis customization	596
MajorTickLines and MinorTickLines customization	596
Tick placement	597
Label format	597
Grouping separator	599
Custom label format	599
Label placement	600
Opposed position	601
Category label	601
Category label customization	602
Working with data	603
Local data	603
Ranges	604
Color customization	605
Actual bar	606
Types of actual bar	607
Actual bar customization	608
Target bar	610
Types of target bar	610
Target bar customization	611
Title and subtitle	612
Title	612
Subtitle	613
Title and subTitle position	614
Title customization	617
SubTitle customization	618
Customization	619
Orientation	619
Right-to-left (RTL)	620
Animation	621
Theme	622
Data label	622
Data label customization	623

Tooltip	625
Default tooltip.....	625
Tooltip template	626
Tooltip customization	626
Accessibility in ASP.NET MVC Bullet chart component	626
WAI-ARIA attributes.....	627
Keyboard interaction	627
Ensuring accessibility	627
See also	628
Button Group	628
Getting Started.....	628
Prerequisites	628
Create ASP.NET MVC application with HTML helper.....	628
Install ASP.NET MVC package in the application	628
Add namespace.....	628
Add stylesheet and script resources	629
Register Syncfusion script manager	629
Add ButtonGroup to the project.....	629
Run the application	630
Orientation.....	630
See Also	630
Types and Styles in Button Group Control.....	631
ButtonGroup types	631
ButtonGroup styles	631
See also	632
Selection and Nesting	632
Selection.....	632
Nesting	633
See also	634
Accessibility in Button Group Control.....	634
Keyboard interaction	635
Ensuring accessibility	636
See also	636
Styles and Appearances	636
How To	636

Create ButtonGroup with icons	636
Create ButtonGroup with rounded corner	637
Disable in Button Group Control.....	638
Enable ripple in Button Group Control	638
Enable RTL in Button Group Control	639
Form submit in Button Group Control	639
Initialize ButtonGroup using util function.....	640
Show ButtonGroup selected state on initial render	641
Button	642
Getting Started with ASP.NET MVC Button Control	642
Prerequisites	642
Create ASP.NET MVC application with HTML helper.....	642
Install ASP.NET MVC package in the application	642
Add namespace.....	642
Add stylesheet and script resources	642
Register Syncfusion script manager	643
Add ASP.NET MVC Button control	643
Change Button type	643
See also	643
Types and Styles in Button Control.....	644
Button styles	644
Button types.....	645
Icons	648
Button size	650
See Also	651
Styles and Appearances	651
Accessibility in Button Control	651
WAI-ARIA attributes.....	652
Keyboard interaction	652
Ensuring accessibility	653
See also	653
How To	653
Add link to a Button	653
Create a Block Button	653
Customize Button Appearance	654

Customize input and anchor elements	655
Repeat Button in Button Control	655
Right to Left in Button Control	657
Set the disabled state.....	657
Tooltip for Button Control	658
Migration from Essential JS 1.....	659
Properties.....	659
Methods.....	660
Events.....	660
Calendar	661
Getting Started with ASP.NET MVC Calendar Control	661
Prerequisites	661
Create ASP.NET MVC application with HTML helper	661
Install ASP.NET MVC package in the application	661
Add namespace.....	661
Add stylesheet and script resources	661
Register Syncfusion script manager	662
Add ASP.NET MVC Calendar control.....	662
Setting the value within min and max dates	663
See also	664
Date Range in Calendar Control.....	664
Multi Selection in Calendar Control.....	664
Globalization in ASP.NET MVC Calendar Control.....	665
Right-To-Left	668
Customization in Calendar Control.....	669
Disable weekends	669
Day cell format.....	670
Highlight Weekends	673
See also	674
Calendar Views in Calendar Control	674
View restriction.....	675
Accessibility in Calendar Control.....	675
WAI-ARIA attributes.....	676
Keyboard interaction	677
Ensuring accessibility	678

See also	678
Displaying Islamic Calendar.....	678
Style and appearance in Calendar Component	685
Customizing the background color for the Calendar	685
Customizing the Calendar date elements on hovering.....	685
Customizing the border of date cell grid	685
Customizing the Calendar title.....	686
Customizing the previous and next icon	686
Customizing the footer button	686
Customizing the selected date cell grid	686
Customizing the content header in Calendar	687
How To	687
Set clear button in Calendar Control	687
Show dates of other months.....	688
Select a sequence of dates in Calendar	689
Skip a month in the Calendar	690
Render the Calendar with week numbers	691
Change the first day of the week	691
Customize the calendar day header	692
Card.....	693
Getting Started with ASP.NET MVC Card Control.....	693
Prerequisites	693
Create ASP.NET MVC application with HTML helper.....	693
Add stylesheet.....	693
Add ASP.NET MVC Card control.....	693
Adding a header and content	694
See also	695
Header and Content in Card Control	695
Header.....	695
Content	696
Images and Divider in Card Control	697
Images.....	697
Divider.....	699
See also	700
Action Buttons in Card Control	700

Vertical	701
See also	702
Horizontal Card in Card Control	702
Stacked cards	702
Carousel	703
Getting Started with ASP.NET MVC Carousel Control	703
Prerequisites	703
Create ASP.NET MVC application with HTML helper	703
Install ASP.NET MVC package in the application	703
Add namespace.....	704
Add stylesheet and script resources	704
Register Syncfusion script manager	704
Add ASP.NET MVC Carousel control	704
Populating Items in ASP.NET MVC Carousel control	706
Populating items using carousel item	706
Populating items using data source	707
Selection.....	709
Partial visible slides	711
See also	714
Animations and Transitions	715
Animations	715
Intervals between slides	718
Auto play slides	719
Pause on hover.....	720
Looping slides.....	721
Slide changing events.....	722
Disable touch swiping	724
Swipe Modes.....	725
Navigators and Indicators	726
Navigators	726
Indicators	731
Play button.....	742
Accessibility in Carousel control	746
ARIA attributes.....	747
Keyboard interaction	748

Ensuring accessibility	748
See also	748
Styles and Appearances in ASP.NET MVC Carousel Component	748
CSS Structure in ASP.NET MVC Carousel Component	748
Customizing the indicators	749
Customizing the navigators.....	751
Customizing partial slides size	753
Chart.....	753
Getting Started with ASP.NET MVC Chart Control.....	753
Prerequisites	753
Create ASP.NET MVC application with HTML helper	753
Install ASP.NET MVC package in the application	753
Add namespace.....	754
Add script resources	754
Register Syncfusion script manager	754
Add ASP.NET MVC Chart control	754
Working with Data	755
Local Data.....	755
Remote Data	758
Empty points	758
Lazy loading.....	760
Chart Dimensions.....	762
Size for Container.....	762
Size for Chart.....	763
Category Axis.....	765
Labels Placement	766
Range	766
Indexed category axis.....	767
Numeric Axis	768
Range	769
Range Padding.....	770
Label Format	775
GroupingSeparator	776
Custom Label Format.....	777
DateTime and DateTimeCategory Axis	778

DateTime Axis	778
DateTimeCategory Axis	779
Label Format	784
Custom Label Format	786
Logarithmic Axis	787
Range	788
Logarithmic Base	789
Logarithmic Interval	789
Axis Labels	790
Smart Axis Labels	790
Axis Labels Positioning	794
Multilevel Labels	796
Edge Label Placement	801
Labels Customization	802
Customizing Specific Point	804
Trim using maximum label width	805
Line break support	806
Maximum Labels	807
Axis Customization in ASP.NET MVC Chart Component	808
Axis Crossing	808
Title	809
Title Rotation	810
Tick Lines Customization	810
Grid Lines Customization	811
Multiple Axis	812
Inversed Axis	813
Opposed Position	814
Strip lines	815
Horizontal Strip lines	815
Vertical Striplines	816
Customize the strip line	817
Customize the stripline text	818
Dash Array	818
Recurrence Stripline	819
Size Type	820

Segment Stripline.....	822
Multiple Panes	823
Rows.....	823
Columns	825
Chart Types	828
Line Chart in ASP.NET MVC Charts Component.....	828
Step line chart in ASP.NET MVC Charts component	831
Stacked Line Chart in ASP.NET MVC Charts Component	833
100% Stacked Line in ASP.NET MVC Charts Component	836
Spline in ASP.NET MVC Charts Component	839
Area in ASP.NET MVC Charts Component	841
Range Area in ASP.NET MVC Charts Component	845
Range step area in ASP.NET MVC Charts component	847
Spline Range Area in ASP.NET MVC Charts Component.....	850
Stacked Area in ASP.NET MVC Charts Component.....	853
100% Stacked Area in ASP.NET MVC Charts Component	855
Stacked step area in ASP.NET MVC Charts component.....	858
Step area in ASP.NET MVC Charts component	861
Spline Area in ASP.NET MVC Charts Component.....	863
Column Chart in ASP.NET MVC Charts Component.....	864
Range Column in ASP.NET MVC Charts Component.....	870
Stacked Column in ASP.NET MVC Charts Component	873
100% Stacked Column in ASP.NET MVC Charts Component	878
Bar Charts in ASP.NET MVC Charts Component	881
Stacked Bar in ASP.NET MVC Charts Component	887
100% Stacked Bar in ASP.NET MVC Charts Component	891
Scatter in ASP.NET MVC Charts Component.....	894
Bubble in ASP.NET MVC Charts Component.....	897
Polar in ASP.NET MVC Charts Component.....	902
Radar in ASP.NET MVC Charts Component.....	913
Hilo in ASP.NET MVC Charts Component.....	917
High Low Open Close in ASP.NET MVC Charts Component.....	919
Candle in ASP.NET MVC Charts Component.....	922
Box and Whisker in ASP.NET MVC Charts Component.....	926
Waterfall in ASP.NET MVC Charts Component.....	931

Histogram in ASP.NET MVC Charts Component	934
Error Bar in ASP.NET MVC Charts Component	937
Vertical Chart in ASP.NET MVC Charts Component	944
Pareto in ASP.NET MVC Charts Component	945
Chart Series in Chart Component	947
Multiple Series	947
Combination Series	949
Enable Complex Property in Series	951
Technical Indicators	951
Accumulation Distribution	952
Average True Range (ATR)	953
Bollinger Band	954
Exponential Moving Average (EMA)	958
Momentum	959
Moving Average Convergence Divergence (MACD)	961
Relative Strength Index (RSI)	963
Simple Moving Average (SMA)	964
Stochastic	965
Triangular Moving Average (TMA)	968
Trend Lines in ASP.NET MVC Chart Component	970
Linear	970
Logarithmic	973
Polynomial	974
Power	975
Moving Average	976
Forecasting	978
Show or hide a trendline	980
Data Markers in ASP.NET MVC Chart Component	981
Marker	982
Shape	982
Images	983
Customization	984
Customizing specific point	985
Fill marker with series color	986
Data Labels in ASP.NET MVC Chart	987

Position	987
Data Label Template	988
Text Mapping	989
Format.....	990
Margin.....	991
Customization	992
Customizing Specific Point	993
Show percentage based on each series points.....	994
Annotation	996
Region	997
Co-ordinate Units.....	998
Alignment.....	999
Adding y-axis sub title through on annotation	1000
Legend and in ASP.NET MVC Chart Component.....	1001
Enable Legend.....	1001
Position and Alignment.....	1002
Legend Reverse	1005
Customization	1007
Series Selection on Legend	1013
Enable Animation.....	1014
Collapsing Legend Item	1016
Legend Title.....	1017
Arrow Page Navigation	1018
Legend Item Padding	1019
Tooltip	1020
Enable tooltip.....	1020
Fixed tooltip	1021
Format the tooltip.....	1022
Tooltip template	1023
Customize the appearance of tooltip	1024
Tooltip mapping name	1025
Zooming and Panning	1026
Enable zooming.....	1026
Modes	1027
Toolbar	1028

Enable pan.....	1029
Enable scrollbar.....	1030
Auto interval on zooming.....	1031
Data Editing.....	1032
Enable Data Editing.....	1032
Crosshair	1033
Tooltip for axis	1034
Customization	1035
Trackball.....	1035
Synchronized charts in ASP.NET MVC Chart Component	1037
Tooltip synchronization.....	1037
Crosshair synchronization	1040
Zooming synchronization.....	1042
Selection synchronization	1045
Selection in ASP.NET MVC Chart Component.....	1047
Point	1048
Series.....	1049
Cluster	1050
Rectangular selection.....	1051
Selection type.....	1054
Selection on load.....	1055
Selection through on legend.....	1056
Customization for selection	1057
Print and Export	1059
Print.....	1059
Export.....	1060
Multiple chart export.....	1065
Appearance in MVC Chart component	1066
Chart theme customization	1066
Custom color palette.....	1067
Data point customization.....	1068
Point level customization.....	1070
Chart area customization.....	1071
Animation.....	1073
Fluid animation	1074

Chart title	1076
Chart subTitle	1079
Rendering Types.....	1080
SVG	1080
Canvas	1080
Accessibility in ASP.NET MVC Chart component	1081
WAI-ARIA attributes.....	1082
Keyboard interaction	1082
Ensuring accessibility	1083
See also	1083
Internationalization.....	1083
Localization	1084
Migration from Essential JS 1.....	1086
Annotations.....	1086
Columns	1087
CommonSeriesOptions	1088
Crosshair	1088
3D chart.....	1089
Canvas rendering	1089
Indicators	1089
Legend.....	1093
primaryXAxis	1096
primaryYAxis	1106
Axes	1116
Rows.....	1124
Series.....	1125
marker.....	1132
TrendLines.....	1135
StripLines.....	1137
Multilevel Labels	1139
Methods.....	1140
Events.....	1141
Chart properties.....	1150
ASP.NET MVC Scaffolding	1151
Getting Started.....	1151

Add a Scaffolded Item	1151
How to render Syncfusion Control'?'	1157
How To	1157
Ajax Chart in ASP.NET MVC Chart control	1157
Customize the background color of data label template from webservice	1161
Render chart in partial view in ajax call	1164
To add dotted line using annotation.....	1166
Hide the tooltip for unselected series	1167
Create a control chart	1168
Add or remove a series from the chart dynamically.....	1179
Render chart from code behind and update the chart using partial view	1181
To add chart dynamically	1185
3D Chart	1186
Getting started with ASP.NET MVC 3D Chart control.....	1186
Prerequisites	1186
Create ASP.NET MVC application with HTML helper	1186
Install ASP.NET MVC package in the application	1186
Add namespace.....	1187
Add script resources	1187
Register Syncfusion script manager	1187
Add ASP.NET MVC 3D Chart control	1187
Working with data in ASP.NET MVC 3D Chart Component	1189
Local data	1189
Remote data.....	1190
Binding data using ODataAdaptor	1190
Empty points	1191
Dimensions in ASP.NET MVC 3D Chart Component	1193
Size for container	1193
Size for chart	1194
Category axis in ASP.NET MVC 3D Chart Component.....	1196
Labels placement	1197
Range	1197
Indexed category axis.....	1198
Numeric axis in ASP.NET MVC 3D Chart Component	1199
Range	1200

Range padding	1201
Label format.....	1207
Grouping separator	1209
Custom label format	1210
DateTime axis in ASP.NET MVC 3D Chart Component	1211
DateTime axis.....	1211
DateTime category axis.....	1212
Label format.....	1217
Custom label format	1218
Logarithmic axis in ASP.NET MVC 3D Chart Component	1219
Range	1220
Logarithmic base	1221
Logarithmic interval	1222
Axis labels in ASP.NET MVC 3D Chart Component	1223
Smart axis labels.....	1223
Edge label placement.....	1226
Maximum labels.....	1227
Axis customization in ASP.NET MVC 3D Chart Component	1228
Title	1228
Title rotation	1229
Tick lines customization	1229
Grid lines customization	1230
Multiple axis.....	1231
Inversed axis.....	1232
Opposed position	1233
Multiple panes in ASP.NET MVC 3D Chart Component	1234
Rows.....	1234
Columns	1237
Chart Types	1239
Column chart in ASP.NET MVC 3D Chart Component	1239
Stacked column chart in ASP.NET MVC 3D Chart Component	1244
100% Stacked column chart in ASP.NET MVC 3D Chart Component	1249
Bar chart in ASP.NET MVC 3D Chart Component	1252
Stacked bar chart in ASP.NET MVC 3D Chart Component	1257
100% Stacked bar chart in ASP.NET MVC 3D Chart Component	1262

Data labels in ASP.NET MVC 3D Chart Component	1265
Position	1266
Template	1267
Text mapping	1268
Format.....	1268
Margin.....	1269
Customization	1270
Customizing specific label	1271
Legend in ASP.NET MVC 3D Chart Component	1272
Position and alignment	1272
Legend customization	1277
Series selection through legend.....	1284
Collapsing legend item.....	1286
Legend title	1287
Arrow page navigation.....	1289
Legend item padding	1290
Tooltip in ASP.NET MVC 3D Chart Component.....	1291
Default tooltip.....	1292
Fixed tooltip	1293
Format the tooltip.....	1294
Tooltip template	1295
Customize the appearance of tooltip	1296
Selection in ASP.NET MVC 3D Chart Component	1297
Point.....	1297
Series.....	1298
Cluster	1299
Selection type.....	1301
Selection during initial loading.....	1302
Selection through legend.....	1303
Print and Export in ASP.NET MVC 3D Chart Component.....	1304
Print.....	1304
Export.....	1305
Appearance in ASP.NET MVC 3D Chart Component.....	1306
Custom color palette.....	1306
Data point customization.....	1308

Point level customization.....	1309
Chart area customization.....	1310
Animation.....	1311
Chart rotation.....	1312
Title	1313
Accessibility in ASP.NET MVC 3D Chart Component	1317
WAI-ARIA.....	1317
Keyboard navigation	1318
CheckBox.....	1318
Getting Started with ASP.NET MVC CheckBox Control.....	1318
Prerequisites	1318
Create ASP.NET MVC application with HTML helper.....	1318
Install ASP.NET MVC package in the application	1318
Add namespace.....	1319
Add stylesheet and script resources	1319
Register Syncfusion script manager	1319
Add ASP.NET MVC CheckBox control.....	1320
Change the CheckBox state	1320
Label and Size in Check Box Control	1320
Label.....	1321
Size	1321
See also	1322
Accessibility in CheckBox Control	1322
WAI-ARIA attributes.....	1323
Keyboard interaction	1323
Ensuring accessibility	1323
See also	1323
Styles and Appearances	1323
How To	1324
Customized CheckBox Control	1324
Name and Value in form submit	1329
Right To Left in Check Box Control.....	1329
Migration from Essential JS 1.....	1330
Properties.....	1330
Methods.....	1331

Events.....	1332
Chips.....	1332
Getting Started with ASP.NET MVC Chip Control	1332
Prerequisites	1332
Create ASP.NET MVC application with HTML helper.....	1333
Install ASP.NET MVC package in the application	1333
Add namespace.....	1333
Add stylesheet and script resources	1333
Register Syncfusion script manager	1334
Add ASP.NET MVC Chip control	1334
Types in Chips Control	1334
Input Chip.....	1334
Choice Chip	1335
Filter Chip.....	1335
Action Chip.....	1335
Chip Customization in Chip Control	1336
Styles	1336
Leading Icon	1337
Avatar.....	1338
Avatar Content.....	1339
Trailing Icon.....	1339
Outline Chip	1339
CSS structures	1340
Customizing the chip text	1340
Customizing the chip icon	1340
Customizing the chip delete button.....	1341
Customizing the chip outline	1341
Customizing the chip on selection	1341
Customizing the chip avatar text	1341
Accessibility in ASP.NET MVC Chips component	1342
WAI-ARIA attributes.....	1343
Keyboard interaction	1343
Ensuring accessibility	1343
See also	1343
CircularGauge.....	1344

Getting Started with ASP.NET MVC Circular Gauge Control.....	1344
Prerequisites	1344
Create ASP.NET MVC application with HTML helper.....	1344
Install ASP.NET MVC package in the application	1344
Add namespace.....	1344
Add stylesheet and script resources	1344
Register Syncfusion script manager	1345
Add ASP.NET MVC Circular Gauge control.....	1345
Add Gauge Title.....	1346
Axis	1346
Circular Gauge Dimensions	1347
Size for Circular Gauge	1347
Axes in Circular Gauge Control	1348
Axis Customization.....	1348
Angles and Direction	1349
Axis Radius	1350
Ticks.....	1351
Labels	1353
Minimum and Maximum	1358
Multiple Axes	1358
Ranges in Circular Gauge Control	1359
Start and End.....	1359
Customization	1360
Radius.....	1360
Dragging Ranges.....	1361
Multiple Ranges	1362
Rounded corner radius	1363
Gradient Color.....	1363
See also	1366
Pointers in Circular Gauge Control.....	1366
Needle Pointers.....	1367
RangeBar Pointer	1369
Rounded corner for range bar pointer	1370
Marker Pointer	1371
Dragging Pointer	1372

Multiple Pointers	1373
Animation.....	1374
Gradient Color.....	1375
Annotations in Circular Gauge Control	1378
Content	1378
Position	1379
Sub Gauge	1380
See also	1384
Animation in Circular Gauge component	1384
Legend in Circular Gauge Control	1385
Legend customization	1385
Position and alignment	1385
Font of the legend text	1386
Toggle option in legend	1387
Paging support in legend	1388
Legend text customization.....	1389
User Interaction in Circular Gauge Control.....	1390
Tooltip for pointers	1390
Tooltip for ranges.....	1392
Tooltip for annotations	1392
Pointer Drag	1393
Print and Export	1394
Print.....	1394
Export.....	1395
Appearance in Circular Gauge Control.....	1398
Gauge Title	1398
Gauge Position	1398
Area Customization.....	1399
Radius calculation based on angles	1402
Accessibility in Circular Gauge component.....	1402
WAI-ARIA attributes.....	1402
Screen reading in Circular Gauge.....	1403
Ensuring accessibility	1403
See also	1403
Internationalization in Circular Gauge Control	1403

Globalization	1403
Right-to-left	1404
Migration from Essential JS 1	1406
Circular gauge dimensions	1406
Axis Line	1406
Ticks	1407
Labels	1409
Ranges	1410
Needle Pointer	1411
Marker Pointer	1412
Rangebar Pointer	1413
Annotations	1413
Appearance	1414
Events	1415
How To	1416
How To	1416
ColorPicker	1420
Getting Started with ASP.NET MVC ColorPicker Control	1420
Prerequisites	1420
Create ASP.NET MVC application with HTML helper	1420
Install ASP.NET MVC package in the application	1420
Add namespace	1421
Add stylesheet and script resources	1421
Register Syncfusion script manager	1421
Add ASP.NET MVC ColorPicker control	1421
Inline type	1422
See also	1423
Mode and Value in Color Picker Control	1423
Rendering palette at initial load	1423
Color value	1424
See also	1425
Localization and RTL	1425
Localization	1425
Right to Left - RTL	1426
See also	1426

Accessibility in Color Picker Control.....	1427
WAI-ARIA attributes.....	1428
Keyboard interaction	1428
Ensuring accessibility	1428
See also	1428
Styles and Appearances	1429
How To	1429
Hide control buttons in Color Picker Control.....	1429
Render palette alone in Color Picker Control	1429
ColorPicker in DropDownButton.....	1430
Customize ColorPicker Control	1432
Handle no color support in Color Picker Control	1440
Disabled in Color Picker Control	1444
Migration from Essential JS 1.....	1444
Properties.....	1444
Methods.....	1446
Events.....	1448
ComboBox.....	1449
Getting Started with ASP.NET MVC ComboBox Control.....	1449
Prerequisites	1449
Create ASP.NET MVC application with HTML helper	1449
Install ASP.NET MVC package in the application	1449
Add namespace.....	1449
Add stylesheet and script resources	1449
Register Syncfusion script manager	1450
Add ASP.NET MVC ComboBox control.....	1450
Custom values.....	1451
Configure the popup list	1451
See also	1452
Data Binding in Combo Box Control	1452
Binding local data.....	1452
Binding remote data	1454
See also	1459
Templates in Combo Box Control	1460
Item template	1460

Group template.....	1461
Header template	1461
Footer template	1462
No records template	1463
Action failure template	1464
See also	1464
Virtualization in ComboBox Component	1465
Binding local data.....	1465
Binding remote data	1466
Grouping	1466
Filtering with Virtualization.....	1467
Grouping in Combo Box Control.....	1468
Customization	1469
Filtering in Combo Box Control.....	1469
Limit the minimum filter character.....	1470
Change the filter type	1471
Case sensitive filtering	1472
Diacritics filtering	1473
See also	1474
Localization in Combo Box Control	1474
Loading translations.....	1474
See also	1475
CSS structures	1475
Customizing the appearance of wrapper element	1475
Customizing the dropdown icon's color	1475
Customizing the focus color	1476
Customizing the outline theme's focus color	1476
Customizing the disabled component's text color	1476
Customizing the float label element's focusing color	1476
Customizing the color of the placeholder text	1477
Customizing the text selection color.....	1477
Customizing the background color of focus, hover, and active item's	1478
Customizing the appearance of pop-up element	1478
Accessibility in Combo Box Control.....	1478
WAI-ARIA attributes.....	1479

Keyboard interaction	1480
Ensuring accessibility	1481
See also	1481
How To	1481
Autofill supported with ComboBox	1481
Configure the Cascading ComboBox.....	1482
Show the list items with icons	1484
ComboBoxFor.....	1488
Migration from Essential JS 1.....	1491
DataBinding.....	1491
Filtering	1491
Template	1492
Applying CSS.....	1493
Grouping	1493
Accessibility.....	1493
Placeholder	1493
Miscellaneous	1494
Sorting.....	1494
Selection.....	1494
Popup.....	1495
Common.....	1495
ContextMenu	1496
Getting Started with ASP.NET MVC ContextMenu Control	1496
Prerequisites	1496
Create ASP.NET MVC application with HTML helper.....	1496
Install ASP.NET MVC package in the application	1496
Add namespace.....	1497
Add stylesheet and script resources	1497
Register Syncfusion script manager	1497
Add ASP.NET MVC ContextMenu control	1498
Rendering items with Separator	1499
See also	1500
Icons and Navigation.....	1500
Icons	1500
Navigation	1503

Template and Multilevel nesting	1505
Template	1505
Multilevel nesting	1506
See Also	1507
Accessibility	1507
WAI-ARIA attributes	1508
Keyboard interaction	1508
Ensuring accessibility	1509
See also	1509
Styles and Appearances	1509
How To	1509
Data Binding in Context Menu Control	1509
Open and close ContextMenu	1510
Template	1512
Underline a character in the item text	1516
Open a dialog on ContextMenu item click	1517
Change animation settings	1518
Dashboard Layout	1520
Getting Started with ASP.NET MVC Dashboard Layout Control	1520
Prerequisites	1520
Create ASP.NET MVC application with HTML helper	1520
Install ASP.NET MVC package in the application	1520
Add namespace	1521
Add stylesheet and script resources	1521
Register Syncfusion script manager	1521
Add ASP.NET MVC Dashboard Layout control	1521
Configuring the layout	1526
Modifying cell size	1526
Setting cell spacing	1528
Graphical representation of the layout	1529
Rendering component in right-to-left direction	1531
Panels	1533
Positioning of panels	1534
Sizing of panels	1536
Header and content of panels	1537

Placing components as content	1540
Adding and removing panels dynamically	1548
Add or remove panels dynamically.....	1548
Dragging of panels	1551
Customizing the dragging handler	1554
Disable dragging of panels	1560
Moving of panels programatically	1562
Resizing of panels.....	1564
Resizing panels programatically.....	1566
Floating panels	1569
Responsive and adaptive layout	1572
Panel state maintenance	1574
CSS structures	1576
Customizing the dashboard layout panel header	1576
Customizing the dashboard layout panel content.....	1576
Customizing the dashboard layout panel resize icon	1577
Customizing the dashboard layout panel background	1577
Accessibility in ASP.NET MVC Dashboard Layout component.....	1577
WAI-ARIA attributes.....	1578
Keyboard interaction	1578
Ensuring accessibility	1579
See also	1579
DatePicker	1579
Getting Started with ASP.NET MVC DatePicker Control	1579
Prerequisites	1579
Create ASP.NET MVC application with HTML helper.....	1579
Install ASP.NET MVC package in the application	1579
Add namespace.....	1579
Add stylesheet and script resources	1580
Register Syncfusion script manager	1580
Add ASP.NET MVC DatePicker control.....	1580
Setting the value within min and max dates	1581
See also	1582
Date Range.....	1582
Date Format	1583

Enable the Masked Input	1584
Configure Mask Placeholder	1585
Globalization in ASP.NET MVC Datepicker Control.....	1586
Right-To-Left	1589
Strict Mode	1590
Customization	1591
See Also	1593
Start and Depth View	1593
Start view	1593
Depth view	1593
Accessibility.....	1594
WAI-ARIA attributes.....	1595
Keyboard Interaction	1595
Ensuring accessibility	1596
See also	1596
Style and appearance in DatePicker Component	1596
Customizing the appearance of DatePicker wrapper element.....	1596
Customizing the DatePicker icon element.....	1597
Customizing the Calendar popup of the DatePicker.....	1597
Full screen mode support in mobiles and tablets.....	1597
How To	1599
Disabled State	1599
Set the placeholder	1599
Customize the datepicker day header	1599
Set the readonly.....	1600
Open the DatePicker popup upon focusing input of DatePicker.....	1601
Prevent the popup close	1602
Client side validation.....	1603
Render DatePickerFor	1604
Posting the selected dates to the controller.....	1605
DateRangePicker	1606
Getting Started with ASP.NET MVC DateRangePicker Control.....	1606
Prerequisites	1606
Create ASP.NET MVC application with HTML helper.....	1606
Install ASP.NET MVC package in the application	1606

Add namespace.....	1607
Add stylesheet and script resources	1607
Register Syncfusion script manager	1607
Add ASP.NET MVC DateRangePicker control.....	1607
Setting the start and end date	1608
See also	1609
Range Restriction	1609
Restrict the range within a range.....	1609
Range span	1610
Strict mode.....	1610
Globalization in ASP.NET MVC DateRangePicker Control.....	1611
Right-To-Left	1614
Customization	1615
Day cell format.....	1615
First day of week	1616
See Also	1616
Accessibility.....	1616
WAI-ARIA attributes.....	1617
Keyboard Interaction	1617
Ensuring accessibility	1619
See also	1619
Style and appearance in DateRangePicker Component	1619
Customizing the appearance of DateRangePicker wrapper element.....	1619
Customizing the DateRangePicker icon element.....	1619
Customizing the DateRangePicker popup calendar header	1620
Customizing the DateRangePicker popup calendar header title	1620
Customizing the DateRangePicker popup calendar content	1620
Customizing the DateRangePicker popup calendar content title.....	1620
Customizing the DateRangePicker popup calendar previous and next icon	1621
Customizing the DateRangePicker popup calendar date cell grid on hovering.....	1621
Customizing the DateRangePicker popup calendar primary button in footer	1621
Customizing the DateRangePicker popup calendar cancel button in footer.....	1622
Customizing the footer element in the DateRangePicker popup calendar	1622
Customizing the selected date cell grid in the DateRangePicker popup calendar	1622
Full screen mode support in mobiles and tablets.....	1622

How To	1624
Disable the control	1624
Set the placeholder	1624
Customization using cssClass	1624
Customize the daterangepicker day header	1626
Render DateRangePickerFor	1627
DateTimePicker	1629
Getting Started with ASP.NET MVC DateTimePicker Control	1629
Prerequisites	1629
Create ASP.NET MVC application with HTML helper	1629
Install ASP.NET MVC package in the application	1629
Add namespace	1630
Add stylesheet and script resources	1630
Register Syncfusion script manager	1630
Add ASP.NET MVC DateTimePicker control	1630
Setting the min and max	1631
See also	1632
Strict Mode	1632
DateTime Range	1634
DateTime Format	1634
Enable the Masked Input	1635
Configure Mask Placeholder	1637
Customization	1638
Day and Time Cell format	1638
See Also	1638
Globalization in DateTimePicker Control	1638
Right-To-Left	1641
Accessibility	1642
WAI-ARIA attributes	1643
Keyboard Interaction	1644
Ensuring accessibility	1645
See also	1646
Style and appearance in DateTimePicker Component	1646
Customizing the appearance of DateTimePicker wrapper element	1646
Customizing the DateTimePicker icons element	1646

Customizing the time picker popup in the DateTimePicker	1646
Customizing the Calendar popup of the DateTimePicker	1647
Full screen mode support in mobiles and tablets.....	1647
How To	1649
Disable the component.....	1649
Customize the datetimepicker day header	1649
Set the placeholder	1650
Render DateTimePickerFor	1650
Diagram.....	1651
Getting Started with ASP.NET MVC Diagram Control.....	1651
Prerequisites	1651
Create ASP.NET MVC application with HTML helper.....	1652
Install ASP.NET MVC package in the application	1652
Add namespace.....	1652
Add stylesheet and script resources	1652
Register Syncfusion script manager	1653
Add ASP.NET MVC Diagram control.....	1653
Connect two Nodes with a Connector	1654
Adding default values	1656
Flow Diagram	1657
Automatic Organization Chart	1660
Map data source	1661
Visualize employee	1663
Node in Diagram Control	1664
Create node.....	1665
Add node through nodes collection.....	1665
Add/Remove node at runtime	1666
Add node from palette.....	1667
Create node through data source.....	1667
Draw nodes	1667
Position	1667
Flip.....	1668
Appearance	1669
Gradient	1670
Linear gradient	1670

Radial gradient	1673
Shadow.....	1676
Customizing shadow	1676
Icon.....	1677
Customizing expand icon	1679
Customizing collapse icon	1679
Interaction.....	1679
Constraints	1679
Custom properties	1679
Stack order	1679
Data flow	1679
See Also	1681
Shapes in Diagram Control.....	1681
Text	1681
Image.....	1682
Image alignment	1684
HTML.....	1685
HTML Node With Template	1686
Native	1686
SVG content alignment	1690
Basic shapes	1691
Path	1692
Flow Shapes	1692
Shapes in Diagram Control.....	1693
Event	1695
Gateway	1698
Activity	1700
Tasks.....	1701
Subprocess	1704
Event subprocess	1705
Transaction subprocess.....	1707
Process	1709
Loop.....	1709
Compensation	1711
Call.....	1712

Adhoc	1714
Boundary	1715
Data	1717
Datasource	1719
Artifact	1720
Text annotation.....	1720
Group	1722
BPMN flows.....	1723
Association	1723
Sequence.....	1725
Message	1726
UML Diagram Shapes	1728
UML Class Diagram	1728
Uml Class Diagram Shapes	1728
UML Class Relationships	1736
How to add UML child at runtime.....	1743
Adding UML Nodes in Symbol palette	1751
Editing	1756
UML Activity diagram.....	1756
UML Activity diagram Shapes	1757
Connector in Diagram	1761
Create connector	1761
Add connectors through connectors collection.....	1761
Add connector at runtime.....	1761
Connectors from palette.....	1763
Draw connectors.....	1763
Update connector at runtime	1763
Connect nodes	1764
Connections with ports	1765
Segments.....	1768
Straight.....	1768
Orthogonal	1769
Avoid overlapping	1771
Bezier	1772
Decorator	1775

Padding	1776
Hit padding.....	1777
Flip.....	1778
Bridging	1779
Corner radius.....	1780
Appearance	1781
Segment appearance	1782
Decorator appearance	1782
Interaction.....	1783
Automatic line routing	1783
Constraints	1785
Custom properties	1786
Stack order	1786
See Also	1787
Group in Diagram Control	1787
Create group	1787
Add group when initializing diagram	1787
Add group at runtime	1790
Add children to group at runtime	1791
Remove children from group at runtime	1792
Container.....	1793
Canvas	1793
Stack.....	1794
Difference between a basic group and containers	1796
Interaction.....	1796
See Also	1796
Swimlane.....	1796
Create a swimlane.....	1796
Lanes	1799
Phase.....	1809
Add swimlane to palette	1813
Limitations.....	1814
Annotation in Diagram.....	1815
Create annotation	1815
Add annotations at runtime.....	1816

Remove annotation	1817
Update annotation at runtime.....	1819
Alignment.....	1820
Offset.....	1820
Horizontal and vertical alignment.....	1821
Annotation alignment with respect to segments	1824
Margin	1826
Text align	1828
Hyperlink	1829
Template Support for Annotation	1831
Wrapping.....	1833
Text overflow	1834
Appearance	1836
Interaction.....	1838
Edit	1839
Read-only annotations.....	1840
Drag Limit	1841
Multiple annotations	1842
Constraints	1843
Ports in Diagram Control.....	1843
Create port.....	1844
Add ports when initializing nodes.....	1844
Add ports at runtime.....	1845
Remove ports at runtime.....	1847
Update port at runtime.....	1849
Appearance	1850
Offset.....	1851
Constraints	1851
Constraints	1851
Diagram constraints.....	1851
Node constraints.....	1852
Connector constraints.....	1853
Port constraints.....	1854
Annotation constraints	1855
Selector constraints	1855

Snap constraints.....	1856
Boundary constraints.....	1857
Inherit behaviors.....	1858
Bitwise operations.....	1859
Add operation.....	1859
Remove Operation.....	1859
Check operation.....	1859
Interaction.....	1859
Bezier segment thumbs.....	1859
End point handles in connector.....	1860
Orthogonal segment thumbs.....	1860
Straight segment editing.....	1861
Selection in diagram.....	1862
Drag and drop nodes over other elements.....	1863
Zoom pan in diagram control.....	1867
Drag Resize and Rotate.....	1868
Keyboard interactions.....	1872
Tools in Diagram Control.....	1872
Drawing tools.....	1872
Shapes.....	1873
Connectors.....	1875
Text.....	1876
Polygon shape.....	1878
Polyline Connector.....	1879
Tool selection.....	1880
Events.....	1881
Freehand Drawing.....	1882
Gridlines.....	1883
Customize the gridlines visibility.....	1884
Appearance.....	1884
Line intervals.....	1885
Snapping.....	1886
Snap to lines.....	1886
Customization of snap intervals.....	1887
Snap to objects.....	1888

Page size and appearance.....	1889
Set background image.....	1889
Multiple page and page breaks.....	1890
Boundary constraints.....	1892
Scroll Settings in Diagram	1893
Get current scroll status.....	1893
Define scroll status.....	1893
Update scroll status	1894
AutoScroll.....	1895
Autoscroll border	1896
Scroll limit	1897
Scroll padding.....	1898
Scrollable Area	1899
UpdateViewport.....	1900
Data Binding in Diagram	1900
Local data	1900
CRUD	1902
Read DataSource.....	1902
How to perform Editing at runtime	1903
InsertData.....	1903
UpdateData	1905
DeleteData	1907
See Also	1909
Automatic Layout in Diagram	1909
Layout modes.....	1909
Hierarchical layout	1909
Radial tree layout.....	1911
Organizational Chart	1913
Symmetric layout	1924
Mind Map layout.....	1925
Tree Orientation in layout.....	1925
Complex hierarchical tree	1927
Customize layout.....	1929
Accessibility.....	1939
WAI-ARIA attributes.....	1940

Aria-label	1940
Keyboard interaction	1941
Ensuring accessibility	1941
See also	1941
Commands	1941
Align	1942
Distribute	1945
Sizing	1947
Clipboard	1949
Grouping	1951
Z-Order command.....	1952
Zoom	1958
Nudge command.....	1960
Nudge by using arrow keys	1961
BringIntoView	1961
BringToCenter	1963
FitToPage command	1964
Command manager.....	1966
Custom command	1966
Modify the existing command	1968
See Also	1969
History List in Diagram	1970
Undo and redo	1970
Undo/redo through shortcut keys	1970
Undo/redo through public APIs	1970
canLog	1972
History change event	1973
Retain Selection	1973
Virtualization in Diagram	1974
Virtualization in Diagram	1974
Serialization in Diagram Control	1974
Save	1974
Load.....	1975
Prevent Default Values	1975
Exporting in Diagram.....	1975

Exporting options	1976
File Name	1976
Format.....	1976
Margin.....	1976
Mode.....	1977
Region	1977
Custom bounds	1978
Export diagram with stretch option.....	1978
Print.....	1979
Tooltip in Diagram Control.....	1980
Default tooltip.....	1980
Common tooltip for all nodes and connectors	1980
Tooltip for a specific node/connector.....	1982
Tooltip for Ports	1982
Tooltip template content.....	1984
Tooltip alignments	1985
Tooltip animation.....	1987
User Handles in Diagram.....	1987
Alignment.....	1987
Fixed user handles	1990
Initialization an fixed user handles	1990
Customization	1991
Customizing the node fixed user handle	1993
Customizing the connector fixed user handle	1995
CSS Structure in Diagram	1999
Customizing the connector end point handle.....	1999
Customizing the connector end point handle when connected.....	1999
Customizing the connector end point handle when disabled	1999
Customizing the bezier connector handle	2000
Customizing the bezier connector line	2000
Customizing the resize handle	2000
Customizing the selector pivot line.....	2000
Customizing the rotate handle	2001
Customizing the symbolpalette while hovering	2001
Customizing the symbolpalette when selected	2001

Customizing the ruler.....	2001
Customizing the ruler overlap.....	2002
Customizing the text edit.....	2002
Customizing the text edit on selection	2002
Ruler	2003
Adding Rulers to the Diagram	2003
Customizing the Ruler	2004
Layers in Diagram Control.....	2006
Visible.....	2007
Lock	2009
Objects	2011
AddInfo.....	2013
Context Menu in Diagram	2017
Customize context menu	2018
Template Support for Context menu.....	2022
Context menu events.....	2023
Symbol Palette in Diagram.....	2025
Create symbol palette.....	2025
Add palettes to SymbolPalette	2026
Customize the palette header	2036
Restrict expansion of the palette panel.....	2046
Stretch the symbols into the palette	2048
Add/Remove symbols to palette at runtime	2050
Customize the size of symbols	2050
Symbol preview.....	2052
Default settings	2054
Adding symbol description for symbols in the palette	2055
Tooltip for symbols in symbol palette	2057
Palette interaction	2060
DragEnter	2060
DragLeave	2060
DragOver	2060
See Also	2060
Overview Control	2060
Create overview	2060

Migration from Essential JS 1.....	2062
Background	2062
Bridging	2063
CommandManager	2063
Connectors	2064
ContextMenu	2077
DataSourceSettings	2080
DefaultSettings.....	2088
DrawType	2089
EnableAutoScroll.....	2090
EnableContextMenu	2090
GetCustomCursor.....	2090
GetCustomProperty	2090
GetDescription	2091
GetCustomTool	2091
Height.....	2091
HistoryManager	2092
Layout.....	2095
Nodes	2099
NodeTemplate	2126
Layers	2126
Annotations.....	2127
PageSettings.....	2131
ScrollSettings.....	2132
SnapSettings.....	2133
ZoomFactor	2135
Tool	2135
ShowTooltip	2136
SelectedItems.....	2136
SerializationSettings.....	2139
How to load EJ1 diagram in EJ2 diagram	2139
Tooltip	2140
How To	2141
HTML Template and CSS in the Organization chart.....	2141
Dialog	2144

Getting Started with ASP.NET MVC Dialog Control	2144
Prerequisites	2144
Create ASP.NET MVC application with HTML helper	2144
Install ASP.NET MVC package in the application	2144
Add namespace.....	2144
Add stylesheet and script resources	2144
Register Syncfusion script manager	2145
Add ASP.NET MVC Dialog control	2145
Modal Dialog	2146
Enable header	2147
Enable footer.....	2148
Draggable	2149
Positioning.....	2149
See also	2150
Template	2151
Header.....	2151
Footer.....	2151
Content	2151
See Also	2155
Animation.....	2155
Resizing	2156
CSS structures	2158
Customizing the dialog header	2158
Customizing the dialog content	2158
Customizing modal dialog overlay	2158
Customizing the dialog resize icon.....	2159
Customizing the dialog close button.....	2159
Customizing the dialog footer button.....	2159
Localization	2160
Loading translations.....	2160
Accessibility.....	2161
WAI-ARIA attributes.....	2162
Keyboard interaction	2162
See Also	2163
Ensuring accessibility	2163

See also	2164
How To	2164
Create nested Dialog.....	2164
Position the Dialog in center of the page on scrolling	2165
Load Dialog content using AJAX.....	2167
Render a Dialog using utility functions	2167
Add minimize and maximize buttons to the Dialog header	2171
Render a Dialog without header	2176
Show Dialog with fullscreen.....	2177
Display a Dialog with custom position	2178
Prevent closing of modal Dialog	2179
Prevent the focus on the first element.....	2181
Prevent opening of the dialog.....	2182
Read all values of Dialog on button click	2185
Customize the Dialog appearance	2188
Close Dialog when click outside of its region.....	2190
Add Icons to Buttons in Dialog Component.....	2191
Setting Maximum Height to the Dialog	2194
Migration from Essential JS 1.....	2195
Accessibility and localization.....	2195
Header.....	2196
Footer.....	2197
Content	2197
Animation.....	2198
Draggable and resizing.....	2198
Target	2199
Position	2199
Visibility.....	2199
Dialog mode	2199
Tooltip	2200
Control state	2200
State maintenance	2200
Common.....	2200
Document Editor	2203
Overview	2203

Key Features.....	2203
Supported Web platforms	2203
Getting Started with ASP.NET MVC DocumentEditor Control.....	2203
Prerequisites	2204
Create ASP.NET MVC application with HTML helper	2204
Install ASP.NET MVC package in the application	2204
Add namespace.....	2204
Add stylesheet and script resources	2204
Register Syncfusion script manager	2205
Add ASP.NET MVC DocumentEditor control.....	2205
See also	2205
Feature modules	2205
Import	2207
Import document from local machine	2208
Convert word documents into SFDT	2209
See Also	2211
Export in Document Editor Component	2211
Sfdt export	2212
Word export.....	2212
Text export	2213
Export as blob	2214
See Also	2216
Collaborative Editing (preview).....	2216
Prerequisites	2216
How to enable collaborative editing in client side.....	2216
How to enable collaborative editing in ASP.NET Core.....	2219
Images	2225
Image resizing	2226
Changing size.....	2227
Text wrapping style	2227
Positioning the image	2227
See Also	2227
Shapes in DocumentEditor	2227
Supported shapes	2227
Text box Shape	2228

Shape Resizer	2228
Text wrapping style	2228
Positioning the shape.....	2228
Text Wrapping Style in DocumentEditor	2228
In-Line with Text.....	2229
In Front of Text.....	2229
Top and Bottom	2229
Behind	2230
Square	2230
Bookmarks	2230
Add bookmark.....	2230
Select Bookmark	2231
Delete Bookmark	2231
Get Bookmark	2231
Bookmark Dialog.....	2231
See Also	2232
Hyperlink.....	2232
Navigate a hyperlink	2232
Copy link.....	2233
Add hyperlink	2234
Customize screen tip.....	2235
Remove hyperlink	2235
Hyperlink dialog	2236
See Also	2236
Tables	2236
Create a table	2236
Insert rows	2237
Insert columns.....	2237
Delete table.....	2238
Delete row.....	2238
Delete column.....	2239
Merge cells.....	2239
Positioning the table	2239
How to work with tables.....	2239
See Also	2247

Table of contents	2247
Inserting table of contents.....	2247
Update or edit table of contents	2249
See Also	2250
Headers and Footers.....	2250
Go to header footer region.....	2250
Link to previous.....	2251
Header and footer distance	2251
Close header footer region	2252
See Also	2252
Working with Text Formatting.....	2252
Bold	2252
Italic.....	2252
Underline property	2253
Strikethrough property	2253
Superscript property	2253
Subscript property	2253
Size	2254
Color.....	2254
Font	2254
Highlight color.....	2254
Toolbar with options for text formatting.....	2254
See Also	2258
Working with Paragraph Formatting	2258
Indentation.....	2258
Special indentation	2258
Increase indent	2258
Decrease indent	2258
Text alignment	2259
Line spacing and its type	2259
Paragraph spacing.....	2259
Pagination properties.....	2259
Paragraph Border	2260
Show or Hide Paragraph marks.....	2261
Toolbar with paragraph formatting options	2261

See Also	2264
Styles	2265
Styles definition overview	2265
Default style	2265
Style hierarchy	2265
Defining new styles	2266
Working with Lists.....	2269
Create bullet list	2269
Create numbered list	2269
Clear list.....	2269
Working with lists	2269
Editing numbered list.....	2270
See Also	2271
Working with Table Formatting	2271
Cell margins.....	2271
Background color	2272
Cell spacing.....	2272
Cell vertical alignment.....	2272
Table alignment	2272
Cell width	2273
Table width	2273
Apply borders.....	2273
Working with row formatting	2274
See Also	2275
Working with Section Formatting.....	2275
Page size.....	2275
Page margins.....	2275
Header distance	2276
Footer distance	2276
See Also	2276
Comments in Document Editor Component	2276
Add a new comment.....	2276
Comment navigation.....	2276
Delete comment	2277
Delete all comment.....	2277

Protect the document in comments only mode.....	2277
Fields	2278
Update fields.....	2278
Get field info	2279
Set field info	2279
See Also	2280
Form Fields in Document Editor Control	2280
Insert form field	2280
Get form field names	2280
Get form field properties	2280
Set form field properties.....	2281
Export form field data	2281
Import form field data	2281
Reset form fields	2282
Protect the document in form filling mode	2282
Clipboard.....	2282
Copy	2282
Cut.....	2283
Paste.....	2283
Local paste	2283
Paste with formatting	2284
See Also	2285
History.....	2285
Enable or disable history.....	2285
Undo and redo	2286
Stack size	2286
See Also	2286
Find and Replace in Document Editor Component	2286
Options pane.....	2286
Search.....	2287
Search results.....	2288
SearchResultsChange event.....	2289
Customize find and replace.....	2289
See Also	2291
Keyboard Shortcuts in Document Editor Component	2291

Text formatting	2291
Paragraph formatting.....	2291
Clipboard	2292
Keyboard shortcut to navigate around the document	2292
Keyboard shortcut to extend selection.....	2292
Find and Replace	2293
Create, Save and Print document	2293
Edit Operation.....	2293
Insert special characters	2293
Dialog	2293
See Also	2294
Scrolling.....	2294
Zooming	2297
Page Fit Type	2298
Zoom option using UI.....	2298
Print in Document Editor Control	2301
Improve print quality	2303
Print using window object	2303
Page setup.....	2303
See Also	2305
Dialog in Document Editor Component.....	2305
Font Dialog	2305
Paragraph dialog	2306
Table dialog	2306
Bookmark dialog	2307
Hyperlink dialog	2308
Table of contents dialog.....	2308
Styles Dialog	2309
Style dialog.....	2309
List dialog	2310
Borders and shading dialog.....	2310
Table options dialog.....	2311
Table properties dialog	2311
Page setup dialog	2312
See Also	2313

Chart.....	2313
Supported Chart Types	2331
Restrict Editing in Document Editor Component.....	2331
Set current user.....	2331
Highlighting the text area	2332
Restrict Editing Pane	2332
Spell Check in Document Editor Component.....	2332
Features	2333
Enable SpellCheck	2333
Disable SpellCheck	2333
Spell check settings	2333
Context menu.....	2335
Globalization	2337
Localization	2337
Document Editor.....	2337
Color Picker	2341
Insert footnote endnote	2342
Insert footnotes	2342
Insert endnotes.....	2342
Update or edit footnotes and endnotes	2343
How To	2344
How to override the keyboard shortcuts in document editor.....	2344
Context menu customization.....	2346
Customize existing toolbar.....	2348
Change document view	2349
How to open a default document in DocumentEditor when initialized.....	2350
How to open a document in read only mode by default in Document Editor	2354
Open a document from URL	2358
Deploy Document Editor component for Mobile	2361
How to disable optimized text measuring in Document Editor component	2361
How to get the selected content in Document Editor component	2362
How to set the default character, paragraph and section format in Document Editor component	2364
How to show and hide spinner while opening document in React Document Editor component	2367
How to change height and width of Document Editor component	2369

How to export the document as PDF in React Document Editor	2370
How to customize the font family drop down in React Document Editor component	2373
How to auto save the document of Document Editor component into AWS S3	2374
How to retrieve the whole document and bookmark content as text in Document Editor component.....	2377
How to select and retrieve the word and paragraph in current cursor position in Document Editor component.....	2381
How to insert page number and navigate to specific page in Document Editor component	2382
How to move the selection to specific position in Document Editor component.....	2384
How to disable header and footer edit in Document Editor component.....	2386
How to insert text, paragraph and rich-text content in Document Editor component.....	2388
How to change the cursor color using CSS in Document Editor component	2390
How to hide the default tool bar and properties pane in Document Editor component.....	2391
How to insert text or image in table programmatically in Document Editor component.....	2392
How to change the default search highlight color in Document Editor component.....	2393
How to optimize the SFDT file	2394
Enable ruler	2395
DropDownButton.....	2397
Getting Started with ASP.NET MVC DropDownButton Control.....	2397
Prerequisites	2397
Create ASP.NET MVC application with HTML helper.....	2397
Install ASP.NET MVC package in the application	2397
Add namespace.....	2397
Add stylesheet and script resources	2397
Register Syncfusion script manager	2398
Add ASP.NET MVC DropDownButton control.....	2398
See also	2399
Icons and Styles in ASP.NET MVC DropDownButton Control	2399
DropDownButton icons.....	2399
See Also	2403
Popup items	2403
Icons	2403
Navigations	2405
Template	2407
See Also	2412

Accessibility in Drop Down Button Component	2412
WAI-ARIA attributes.....	2413
Keyboard interaction	2414
Ensuring accessibility	2414
See also	2414
Styles and Appearances	2414
How To	2415
Change caret icon	2415
Create DropDownButton with rounded corner	2416
Create right-to-left DropDownButton	2417
Customize icon and width.....	2419
Disable a DropDownButton	2420
Group popup items with ListView component.....	2422
Hide dropdown arrow.....	2423
Open a dialog on popup item click	2424
Position popup open.....	2425
Underline a character in the item text.....	2427
DropDownList	2428
Getting Started with ASP.NET MVC DropDownList Control	2428
Prerequisites	2428
Create ASP.NET MVC application with HTML helper	2428
Install ASP.NET MVC package in the application	2428
Add namespace.....	2428
Add stylesheet and script resources	2428
Register Syncfusion script manager	2429
Add ASP.NET MVC DropDownList control	2429
Binding data source	2430
Configure the popup list	2430
See also	2431
Data Binding.....	2431
Binding local data.....	2431
Binding remote data	2434
See Also.....	2439
Templates.....	2439
Item template	2439

Value template.....	2440
Group template.....	2441
Header template	2442
Footer template	2443
No records template	2444
Action failure template	2444
See Also	2445
Virtualization in DropDown List	2445
Binding local data	2446
Binding remote data	2446
Grouping	2447
Filtering with Virtualization.....	2448
Grouping	2449
Customization	2450
Filtering	2450
Limit the minimum filter character	2451
Change the filter type	2452
Case sensitive filtering	2453
Diacritics Filtering.....	2454
See Also	2454
Localization in ASP.NET MVC DropDownList control.....	2455
Loading translations.....	2455
See Also	2456
CSS structures	2456
Customizing the appearance of wrapper element	2456
Customizing the dropdown icon's color	2456
Customizing the focus color	2456
Customizing the outline theme's focus color	2457
Customizing the disabled component's text color	2457
Customizing the float label element's focusing color	2457
Customizing the color of the placeholder text	2458
Customizing the placeholder to add mandatory indicator(*).....	2458
Customizing the background color of focus, hover, and active item	2458
Customizing the appearance of pop-up element	2458
Accessibility.....	2459

WAI-ARIA attributes.....	2460
Keyboard interaction	2460
Ensuring accessibility	2461
See also	2461
How To	2462
Add item in between in DropDownList.....	2462
Configure the Cascading DropDownList	2463
Preselect value from model in Cascading DropDownList	2464
Model binding with nested class	2466
Clear the selected item in DropDownList	2467
Close the popup on scroll.....	2468
Disable the Fixed group header in DropDownList	2469
Highlight the matched character in filtering.....	2470
Show the list items with icons	2471
Do incremental search in the DropDownList.....	2475
Modify the result data before passing to DropDownList when binding remote data source	2475
Preselect the list items in multiple cascading DropDownList	2476
Get the total count of data when remote data bind with DropDownList	2478
Remove an item from DropDownList	2479
Limit the search result on filtering.....	2480
DropDownList options with tooltip.....	2481
Detect whether the value change happened by manual or programmatic	2482
Whether each list items hold unique value	2484
DropDownListFor	2484
Migration from Essential JS 1.....	2486
DataBinding.....	2486
Filtering	2487
Template	2488
Virtual Scrolling.....	2488
Applying CSS.....	2488
Sorting	2489
Popup	2489
Placeholder	2490
Grouping	2491
Accessibility.....	2491

Miscellaneous	2491
Selection.....	2492
Common.....	2493
DropDownTree.....	2494
Getting Started with ASP.NET MVC DropDownTree Control.....	2494
Prerequisites	2494
Create ASP.NET MVC application with HTML helper	2494
Install ASP.NET MVC package in the application	2494
Add namespace.....	2494
Add stylesheet and script resources	2494
Register Syncfusion script manager	2495
Add ASP.NET MVC DropDownTree control	2495
Binding data source	2495
Data Binding in Dropdown Tree Component.....	2497
Local data	2498
Remote data.....	2505
Prevent Node selection.....	2507
Templates in Drop Down Tree Component	2511
Item template	2511
Header template	2513
Footer template	2515
No records template	2517
Action failure template	2518
Custom template to show selected items in input	2520
CheckBox.....	2524
Auto Check	2528
Select All.....	2531
Accessibility in ASP.NET MVC Dropdown Tree component.....	2535
WAI-ARIA attributes.....	2536
Keyboard interaction	2537
Ensuring accessibility	2537
See also	2538
Localization in Dropdown Tree Component	2538
File Manager	2538
Getting Started with ASP.NET MVC FileManager Control	2538

Prerequisites	2538
Create ASP.NET MVC application with HTML helper	2538
Install ASP.NET MVC package in the application	2538
Add namespace.....	2539
Add stylesheet and script resources	2539
Register Syncfusion script manager	2539
Add ASP.NET MVC FileManager Control.....	2539
File Download support.....	2543
File Upload support.....	2543
Image Preview support	2543
File Manager Overview	2544
Switching initial view of the File Manager	2547
Maintaining component state on page reload	2550
Rendering component in right-to-left direction	2554
Specifying the current path of the File Manager	2557
User Interface Structure	2560
Toolbar	2561
Files and folders navigation	2562
View	2563
Context menu.....	2564
File Operations in FileManager Component.....	2565
Folder Upload support	2566
File operation request and response Parameters	2572
File request and response contents.....	2589
Action Buttons	2590
Views.....	2591
Largelcons View	2591
Details View.....	2594
Customizing File Manager functionalities.....	2597
Context menu customization.....	2597
Details view customization	2600
Navigation pane customization	2604
Show/Hide file extension	2607
Show/Hide hidden items.....	2610
Show/Hide thumbnail images in large icons view	2613

Toolbar customization	2616
Upload customization	2620
Tooltip customization	2623
Multiple Selection	2628
Drag And Drop	2631
File system provider	2634
ASP.NET Core file system provider	2635
ASP.NET MVC 5 file system provider	2636
ASP.NET Core Azure cloud file system Provider	2636
ASP.NET MVC 5 Azure cloud file system Provider	2638
Amazon S3 cloud file system provider	2639
File Transfer Protocol file system provider	2640
SQL database file system provider	2641
NodeJS file system provider	2642
Google Drive file system provider	2644
Firebase Realtime Database file system provider	2645
IBM Cloud Object Storage file provider	2651
Localization	2652
Virtualization in File Manager Component	2660
Enable Virtualization	2660
Limitations for Virtualization	2664
Accessibility in ASP.NET MVC File Manager component	2664
WAI-ARIA attributes	2665
Keyboard interaction	2666
Ensuring accessibility	2667
See also	2667
Access Control in File Manager component	2667
Access Rules	2668
Permissions	2669
How To	2670
How to add custom menu item in context menu	2670
How to add custom button in toolbar	2674
How to enable/disable toolbar item/items	2678
How to add custom thumbnail in file manager	2682
Nested FileManager	2701

Render FileManager in Internet Explorer	2713
Create the custom file provider using NodeJS.....	2715
Floating Action Button	2734
Getting Started with ASP.NET MVC Floating Action Button Control	2734
Prerequisites	2734
Create ASP.NET MVC application with HTML helper.....	2734
Install ASP.NET MVC package in the application	2734
Add namespace.....	2735
Add stylesheet and script resources	2735
Register Syncfusion script manager	2735
Add ASP.NET MVC Floating Action Button control	2736
Event Click In Floating Action Button.....	2736
Icons in Asp.Net MVC Floating Action Button Control	2736
FAB with icon	2737
FAB with icon and text.....	2738
Styles in ASP.NET MVC Floating Action Button Control.....	2742
FAB styles	2742
Styles customization	2743
Show text on hover	2743
Positions in Asp.Net MVC Floating Action Button Control	2744
Custom position	2748
Events in Floating Action Button Control.....	2749
Created.....	2749
OnClick	2749
Gantt	2750
Getting Started with ASP.NET MVC Gantt Control	2750
Prerequisites	2750
Create ASP.NET MVC application with HTML helper.....	2750
Install ASP.NET MVC package in the application	2750
Add namespace.....	2750
Add stylesheet and script resources	2750
Register Syncfusion script manager	2751
Add ASP.NET MVC Gantt Control.....	2751
Defining columns	2754
Enable editing	2755

Enable filtering	2759
Enable sorting	2760
Enabling predecessors or task relationships	2760
Assigning Resources	2760
Data Binding in Gantt	2764
Local data	2765
Remote data	2769
Split task	2785
Improve performance by disabling validations	2787
Limitations	2788
Columns in in gantt control	2788
Defining columns	2788
Custom column header	2790
Format	2791
Change tree/expander column	2793
Show or hide columns dynamically	2794
Controlling gantt column actions	2795
Column type	2795
Timeline	2796
Timeline view modes	2796
Timeline cells tooltip	2801
Splitter in ASP.NET MVC Gantt component	2802
Splitter	2802
Change splitter position dynamically	2803
Task Scheduling	2804
Automatically scheduled tasks	2804
Manually scheduled tasks	2808
Custom	2813
Unscheduled tasks	2817
Define unscheduled tasks in data source	2818
Working time range	2821
Weekend/Non-working days	2831
Duration units	2840
Taskbar in ASP.NET MVC Gantt Chart Component	2845
Taskbar template	2845

Taskbar customization	2847
Connector lines	2851
Tooltip	2854
Tooltip template	2858
Critical Path feature	2864
Customize taskbar in critical path.....	2865
Toolbar in Gantt control	2866
Built-in toolbar items	2866
Custom toolbar items	2867
Built-in and custom items in toolbar	2868
Enable/disable toolbar items	2869
Add input elements to toolbar.....	2870
Managing Tasks in ASP.NET MVC Gantt Component	2870
Cell edit type and its params.....	2871
Cell edit template.....	2872
Disable editing for particular column	2873
Read-only gantt.....	2874
Open new task dialog with default values	2874
Troubleshoot: Editing works only when primary key column is defined.....	2878
Touch interaction	2878
Taskbar editing tooltip	2880
Scrolling in Gantt Control.....	2881
Set width and height	2881
Responsive with the parent container.....	2882
Scroll To Date method	2882
Set the vertical scroll position.....	2883
Virtual Scrolling in Gantt	2884
Row virtualization	2884
Timeline virtualization	2885
Limitations for virtual scroll	2886
Resources in Syncfusion ASP.NET MVC Gantt Component	2886
Resource collection	2886
Assign resource	2890
Add/Edit resource collection	2896
Resource View in ASP.NET MVC Gantt Component	2897

Resource task	2897
Resource OverAllocation.....	2902
Unassigned task	2907
Enable taskbar drag and drop.....	2907
Filtering in gantt control	2912
Filter hierarchy modes	2912
Filtering a specific column by method	2916
Clear filtered columns	2916
Sorting in ASP.NET MVC Gantt Chart Component.....	2917
Sorting column on Gantt initialization	2918
Sorting column dynamically	2919
Clear all the sorted columns dynamically	2920
Sorting events	2921
Sorting custom columns.....	2922
Touch interaction	2924
Selection.....	2925
Selection mode	2926
Toggle selection	2926
Clear selection.....	2927
Get selected row indexes and records.....	2928
Multiple selection based on condition	2928
Touch interaction	2929
See Also	2930
Rows in ASP.NET MVC Gantt Component	2930
Row height	2930
Expand/Collapse row	2931
Customize rows.....	2935
Styling alternate rows	2936
Row spanning.....	2937
Customize rows and cells.....	2938
Clip mode	2940
Export.....	2941
Excel Export in gantt control.....	2941
Export.....	2943
Context menu in ASP.NET MVC Gantt component.....	2955

Custom context menu items	2957
Touch interaction	2959
State Persistence Feature	2959
Get or set localStorage value	2959
Prevent columns from persisting	2960
Persist the header template and header Text	2961
Accessibility	2962
WAI-ARIA attributes	2963
Keyboard navigation	2964
Ensuring accessibility	2965
See also	2965
Globalization in Gantt control	2965
Localization	2965
Internationalization	2969
Right to left (RTL)	2970
See Also	2972
Styling	2973
Grid lines	2974
Timezone	2975
Understanding date manipulation in JavaScript	2975
Display same time everywhere with no time difference	2975
CRUD operations with timezone	2978
Timezone methods	2981
Migration from Essential JS 1	2983
Data Binding and Task mapping	2983
Sorting	2984
Filtering	2985
Searching	2986
Selection	2986
Editing	2988
Columns	2991
Toolbar	2992
ToolTip	2992
Timeline	2994
Rows	2995

Resources	2997
Baseline	2997
Context Menu	2997
Scheduling Tasks	2998
Appearance and Customizations	2999
Stripline	3001
Holidays.....	3001
Others	3002
How To	3004
Open add/edit dialog dynamically	3004
Set the vertical scroll position.....	3005
Change schedule start and end dates.....	3006
Copy and Paste records in Gantt.....	3007
Updating row drag and dropped index position on server side	3008
Add Custom Fields in the General Tab in Add/Edit Dialog.....	3010
Maintain zoomToFit	3012
Drag and Drop the Record from another component to Gantt.....	3018
Set new row position in Gantt	3020
Open add/edit dialog dynamically	3021
Grid.....	3022
Getting Started with ASP.NET MVC Grid Control	3022
Prerequisites	3022
Create ASP.NET MVC application with HTML helper	3022
Install ASP.NET MVC package in the application	3022
Add namespace.....	3023
Add stylesheet and script resources	3023
Register Syncfusion script manager	3023
Add ASP.NET MVC Grid control	3023
Defining Row Data	3024
Defining Columns	3025
Enable Paging.....	3026
Enable Sorting	3027
Enable Filtering	3028
Enable Grouping.....	3029
See also	3030

Data Binding.....	3030
Sending additional parameters to the server	3030
Handling HTTP error.....	3031
Binding with ajax.....	3032
Troubleshoot: Grid render rows without data	3032
See Also	3033
Data Annotation in Grid Component	3033
Immutable Mode	3035
Limitations.....	3035
Performance tips for ASP.NET MVC Grid control	3035
How to improve loading performance by binding large dataset.....	3036
How to improve loading performance by binding large data by showing custom text or element	3036
How to improve loading performance by referring individual script and CSS	3037
How to update cell values without frequent server calls	3037
How to optimize server-side data operations with adaptors	3037
How to avoid MaxJsonLength error while passing large amount of records	3037
Microsoft excel limitation while exporting millions of records to excel file format.....	3038
Columns in ASP.NET MVC Grid Component	3038
Column types	3038
ValueAccessor	3039
Format.....	3039
Render boolean value as checkbox.....	3041
Visibility	3042
Lock columns.....	3042
Controlling Grid actions	3043
Show or hide columns by external button.....	3043
Customize column styles.....	3044
Display custom tooltip for columns	3046
Align the text of Grid content and header	3047
How to prevent checkbox in the blank row	3047
See Also	3048
Row in ASP.NET MVC Grid Control	3048
Row customization.....	3049
Adding a new row programmatically.....	3051

How to get the row information when hovering over the cell	3052
See Also	3053
Cell	3053
Cell customization	3053
Custom attributes	3054
Grid lines	3055
Editing in ASP.NET MVC Grid Component	3055
Disable editing for particular column	3057
Disable editing for a particular row or cell.....	3057
Editing template column.....	3059
Troubleshoot editing works only for first row	3059
How to make a Grid column always editable	3059
See Also	3061
Sorting.....	3061
Initial sort	3061
Multi-column sorting	3062
Sort order	3063
Sort foreign key column based on Text	3063
Sorting events	3064
Touch interaction	3065
See Also	3066
Grouping in ASP.Net MVC Grid Component	3066
Initial group	3067
Hide drop area	3067
Group with paging.....	3068
Group by format	3068
Grouping events.....	3069
Collapse by external button	3069
Sort grouped columns in descending order during initial grouping	3070
See Also	3071
Filtering in ASP.Net MVC Grid Component.....	3071
Initial filter	3072
Filter operators	3073
Wildcard and LIKE operator filter	3073
See Also	3074

Search in ASP.NET MVC Grid Component.....	3074
Initial search	3074
Search operators.....	3075
Search by external button.....	3075
Search specific columns	3076
Clear search by external button.....	3077
Search on each key stroke	3077
Perform search operation in Grid using multiple keywords.....	3078
See Also	3080
Paging.....	3080
Template	3081
Pager with Page Size Dropdown	3082
How to render Pager at the Top of the Grid.....	3083
See Also	3083
Scrolling.....	3084
Set width and height.....	3084
Responsive with parent container	3084
Scroll to selected row.....	3085
Hide the scrollbar when the content is not overflown	3086
Frozen rows and columns	3087
Limitations of Frozen Grid.....	3087
Freeze particular columns.....	3088
Freeze Direction	3088
Deprecated Methods	3089
Virtualization in ASP.NET MVC Grid Component.....	3091
Row Virtualization.....	3091
Column Virtualization	3092
Virtualization with Grouping.....	3096
Limitations for virtual scrolling	3096
Browser height limitation in virtual scrolling and solution.....	3097
Infinite scrolling in ASP.NET MVC Grid Component.....	3104
InitialBlocks	3105
Cache Mode	3107
Limitations for Infinite Scrolling.....	3109
Selection.....	3109

Selection mode	3110
Touch interaction	3111
Aggregates	3111
Built-in aggregate types	3111
Print in ASP.NET MVC Grid Component.....	3111
Page setup.....	3112
Print using an external button	3112
Print the visible page.....	3113
Print the hierarchy grid	3113
Print the master detail grid	3115
Print large number of columns	3116
Show or Hide columns while Printing	3117
Limitations of Printing Large Data.....	3118
See Also	3118
Adaptive View	3118
Render adaptive dialogs.....	3118
Vertical row rendering	3119
Scaffolding.....	3120
Add a Scaffolded Item	3120
How to render Syncfusion Control'?'	3125
State Persistence.....	3125
Restore initial Grid state	3125
Maintaining custom query in a persistent state	3126
Add a new column in persisted columns list	3126
ToolBar in ASP.NET MVC Grid Component.....	3127
Enable or disable toolbar items	3127
Add toolbar at the bottom of Grid.....	3128
PDF Export.....	3129
Show spinner while exporting.....	3130
Custom data source	3131
Passing additional parameters to the server when exporting.....	3132
Export large number of columns in a single page.....	3133
See Also	3134
Excel Export.....	3134
Show spinner while exporting.....	3135

Custom data source	3136
Passing additional parameters to the server when exporting.....	3137
See Also	3138
Globalization in ASP.NET MVC Grid control.....	3138
Localization	3138
Internationalization.....	3143
Right to left (RTL)	3144
See Also	3145
Accessibility in Grid component	3145
WAI-ARIA attributes.....	3145
Keyboard interaction	3146
Ensuring accessibility	3147
See also	3148
Clipboard.....	3148
Copy to clipboard by external buttons	3148
AutoFill	3149
Paste.....	3150
Context menu in ASP.NET MVC Grid Component	3151
Custom context menu items.....	3152
Show context menu on left click.....	3153
Enable or disable context menu items	3154
Loading Animation ASP.NET MVC Grid Component.....	3155
Style and Appearance	3156
Customizing the Grid root element	3156
How To	3157
Migration from Essential JS 1.....	3157
Enable/Disable Grid and its actions	3194
Print the expanded state from other pages.....	3196
Perform CRUD operation using anti-forgery token	3198
Perform Grid actions by keyboard shortcut keys	3202
Customize Pager DropDown	3203
Hide the expand/collapse icon in parent row with no record in child grid	3204
Render both EJ1 and EJ2 Grids in same application	3206
Set complex column as Foreignkey column.....	3207
Complex Data Binding with list of Array Of Objects	3208

Select grid rows based on certain condition.....	3208
Collapse all grouped rows at initial render	3209
How to show grouped rows based on the pageSize	3210
Get specific row and cell index in Grid.....	3211
Display the null date values at bottom of the Grid while perform sorting.....	3211
Avoid TypeScript Compilation.....	3213
Enable editing in single click	3213
Cascading DropDownList with Grid editing	3215
Hide sorting options on excel filter Dialog.....	3217
Add a title to the header when using Grid print action	3218
Customizing Filter Dialog by using an additional Parameter	3218
Customize the Empty Record Template	3219
HeatMap Chart.....	3221
Getting Started with ASP.NET MVC HeatMapChart Control	3221
Prerequisites	3221
Create ASP.NET MVC application with HTML helper	3221
Install ASP.NET MVC package in the application	3221
Add namespace.....	3221
Add stylesheet and script resources	3222
Register Syncfusion script manager	3222
Add ASP.NET MVC HeatMapChart control	3222
Populate heat map with data	3222
Enable axis labels	3223
Add heat map title	3225
Enable legend.....	3226
Add data label	3227
Add custom cell palette	3228
Enable tooltip.....	3229
Working with data in ASP.NET MVC HeatMap Chart Component	3230
Data adaptor	3230
Empty points	3237
Binding nested JSON data	3238
See Also	3240
Bubble HeatMap in ASP.NET MVC HeatMap Chart Component	3240
Bubble types	3241

Rendering mode in ASP.NET MVC HeatMap Chart Component.....	3256
Axis in ASP.NET MVC HeatMap Chart Component.....	3257
Types	3257
Inversed axis.....	3261
Opposed axis.....	3262
Axis labels customization	3263
Axis intervals	3269
Axis label increment.....	3270
Limiting labels in date-time axis.....	3271
Multilevel Labels	3274
Palette in ASP.NET MVC HeatMap Chart Component.....	3276
Palette types	3276
Defining color stops	3278
See Also	3280
Legend in ASP.NET MVC HeatMap Chart Component.....	3280
Legend types	3281
Placement	3282
Alignment.....	3283
Legend dimensions	3284
Paging for legend	3285
Smart Legend	3287
Legend Selection	3288
Legend Title.....	3289
Appearance in ASP.NET MVC HeatMap Chart Component.....	3290
Cell customization.....	3290
Background color	3294
Margin.....	3295
Title	3296
Data label	3297
Dimensions in ASP.NET MVC HeatMap Chart Component.....	3305
Size for container	3305
Size for heat map	3305
In pixel.....	3305
In percentage	3306
Tooltip in ASP.NET MVC HeatMap Chart Component.....	3307

Default tooltip.....	3307
Tooltip template	3309
Customize the appearance of Tooltip.....	3310
Selection in ASP.NET MVC HeatMap Chart Component.....	3311
Enable single cell selection	3312
Accessibility in ASP.NET MVC HeatMap chart component.....	3315
Ensuring accessibility	3315
See also	3315
Events in ASP.NET MVC HeatMap Chart Component.....	3315
CellClick	3315
CellDoubleClick	3316
CellRender	3317
CellSelected.....	3318
Created.....	3319
LegendRender	3320
Load.....	3322
Loaded.....	3323
Resized	3324
TooltipRender	3325
How To	3326
Customizing tooltip as a table.....	3326
Change the legend label text	3327
Migration from Essential JS 1.....	3328
Members.....	3328
Events.....	3331
In-place Editor	3332
Getting Started with ASP.NET MVC In-place Editor Control.....	3332
Prerequisites	3332
Create ASP.NET MVC application with HTML helper.....	3332
Install ASP.NET MVC package in the application	3332
Add namespace.....	3332
Add stylesheet and script resources	3332
Register Syncfusion script manager	3333
Add ASP.NET MVC In-place Editor control.....	3333
Add the In-place Editor with Textbox	3333

Configuring DropDownList.....	3334
Integrate DatePicker	3335
Submitting data to the server (save)	3337
Refresh with modified value	3337
See also	3339
List of controls.....	3339
Model configuration	3344
See Also	3344
Configuration	3344
Rendering modes	3344
Event actions for editing	3348
Action on focus out	3350
Display modes	3351
Buttons.....	3353
See Also	3355
Server actions.....	3355
See Also	3357
Data Binding.....	3357
Local	3357
Remote.....	3359
Integrate HTML5 controls (Template)	3362
As a string.....	3362
As a selector	3362
Validation.....	3363
Validation Rules	3364
CSS structures	3364
Customizing the In-place Editor text.....	3364
Customizing the In-place Editor action buttons	3365
Globalization	3365
Localization	3365
Right to left	3367
Format.....	3368
How To	3369
Dynamically move input to edit mode.....	3369
Disable the edit mode specifically	3371

Add custom indication to unsaved value.....	3372
Set custom animation for popup mode.....	3374
Image Editor.....	3375
End User Capabilities in the ASP.NET MVC Image Editor control.....	3375
Open an image.....	3375
Zooming.....	3376
Panning.....	3377
Cropping and image transformation.....	3378
Annotations.....	3379
Filtering and fine-tune.....	3380
Undo and redo the operations.....	3381
Reset an image.....	3382
Export an image.....	3383
Getting Started with ASP.NET MVC Image Editor Control.....	3384
Prerequisites.....	3384
Create ASP.NET MVC application with HTML helper.....	3384
Install ASP.NET MVC package in the application.....	3384
Add namespace.....	3385
Add stylesheet and script resources.....	3385
Register Syncfusion script manager.....	3385
Add ASP.NET MVC Image Editor control.....	3385
Open and Save in the ASP.NET MVC Image Editor control.....	3386
Open.....	3387
Supported image formats.....	3388
File opened event.....	3390
Saving event.....	3390
Created event.....	3391
Destroyed event.....	3391
Reset an image.....	3391
Selection cropping in the ASP.NET MVC Image Editor control.....	3391
Insert custom / square / circle region.....	3391
Insert selection based on aspect ratio.....	3393
Crop an image.....	3395
Cropping event.....	3396
Annotation in the ASP.NET MVC Image Editor control.....	3397

Text annotation.....	3397
Freehand drawing	3406
Shape annotation.....	3410
Delete a shape	3413
Image annotation.....	3414
Transform in the ASP.NET MVC Image Editor control	3416
Rotate an image	3416
Flip an image	3418
Straighten an image	3419
Zoom in or out an image.....	3421
Panning an image.....	3425
Zooming event	3427
Rotating event.....	3427
Flipping event.....	3428
Toolbar in the ASP.NET MVC Image Editor control	3428
Built-in toolbar items	3428
Add a custom toolbar items.....	3428
Show or hide a toolbar.....	3430
Show or hide a toolbar item	3431
Enable or disable a toolbar item	3433
Enable or disable a contextual toolbar item.....	3433
Toolbar created event.....	3433
Toolbar item clicked event.....	3433
Toolbar template	3435
Customize contextual toolbar.....	3436
Quick access toolbar in the ASP.NET MVC Image Editor control.....	3438
Add a custom toolbar item	3438
Undo Redo in the ASP.NET MVC Image Editor control.....	3439
Undo the action	3439
Redo the Action	3440
Filters in the ASP.NET MVC Image Editor control	3441
Apply filter effect	3442
Finetune in the ASP.NET MVC Image Editor control.....	3444
Adjust the brightness, contrast, or sharpness	3444
Adjust the hue, exposure, blur, or opacity	3446

Finetune value changing event	3448
Frames.....	3449
Apply frame to the Image	3449
Frame changing event.....	3451
Resize	3452
Apply resize to the image.....	3452
Resizing event	3453
Accessibility in Image editor control.....	3453
Keyboard interaction	3454
Ensuring accessibility	3455
See also	3455

Syncfusion ASP.NET MVC UI (Essential JS 2)

The Syncfusion ASP.NET MVC UI (Essential JS 2) is a modern enterprise UI toolkit that has been built from the ground up to be lightweight, responsive, modular and touch friendly. It also available in other frameworks such as JavaScript and Angular, React.

How to best read this user guide

- The best way to get started would be to read the "Getting Started" section of the documentation for the component that you would like to start using first. The "Getting Started" guide gives just enough information that you need to know before starting to write code. This is the only section that we recommend reading end-to-end before starting to write code, all other information can be referred as needed.
- Now that you are familiar with the basics of using the component, the next step would be to start integrating the component into your application. A good starting point would be to refer to the code snippets in the [online sample browser](#) which contains hundreds of code samples, it is very likely that you will find a code sample that resembles your intended usage scenario.
- Another valuable resource is the API reference which provides detailed information on the object hierarchy as well as the settings available on every object.

Control List

<style>

table

```
{
border:0 !important;
line-height: 160% !important;
}
tr
{
border:0 !important;
}
td
{
border:0 !important;
vertical-align: top;
}
.controlanchorlink
{
font-size: 14px !important;
```

```
text-decoration: none!important;
```

```
text-align: left!important;
```

```
padding: 1px 0px;
```

```
}
```

```
.controlcategory-topics
```

```
{
```

```
font-size: 14px !important;
```

```
font-weight: 500!important;
```

```
border:0 !important;
```

```
line-height: 20px;
```

```
}
```

```
.controlcategory
```

```
{
```

```
font-size: 14px !important;
```

```
font-weight: 500!important;
```

```
border:0 !important;
```

```
text-align: left!important;
```

```
line-height: 20px;
```

```
padding-top: 20px;
```

```
}
```

```
<style>
```

table

```
{
```

```
border:0 !important;
```

```
line-height: 160% !important;
```

```
}
```

```
tr
```

```
{
```

```
border:0 !important;
```

```
}
```

```
td
```

```
{
```

```
border:0 !important;
```

```
vertical-align: top;
}
.controlanchorlink
{
font-size: 14px !important;
text-decoration: none!important;
text-align: left!important;
padding: 2px 0px;
}
.controlcategory-topics
{
font-size: 14px !important;
font-weight: 500!important;
border:0 !important;
line-height: 20px;
}
.controlcategory
{
font-size: 14px !important;
font-weight: 500!important;
border:0 !important;
text-align: left!important;
line-height: 20px;
padding-top: 50px;
}
</style>
```

| | | | |
|-----------------------------|--------------------------------|---------------------------------|--------------------------------------|
| GRIDS | DATA
VISUALIZATION | CALENDARS | DROPDOWNS |
| DataGrid | Charts | Scheduler | AutoComplete |
| Pivot Table | Stock Chart | Gantt Chart | ListBox |
| TreeGrid | Circular Gauge | Calendar | ComboBox |
| Spreadsheet | Linear Gauge | DatePicker | Dropdown List |
| | Diagram | DateRangePicker | Multiselect DropDown |
| | | | |

| | | | |
|--------------------------------|--|--------------------------------------|------------------------------|
| FILE VIEWERS & EDITORS | HeatMap Chart | DateTime Picker | Mention |
| | Map | TimePicker | NAVIGATION |
| RichTextEditor | Smith Chart | INPUTS | Accordion |
| PDF Viewer | Sparkline Charts | TextBox | Context Menu |
| Word Processor | Barcode | Input Mask | Menu Bar |
| Image Editor | TreeMap | Numeric TextBox | Sidebar |
| FILE FORMAT FRAMEWORKS | Bullet Chart | RadioButton | Tabs |
| | Kanban | CheckBox | Toolbar |
| | BUTTONS | Color Picker | TreeView |
| Excel | Button | File Upload | File Manager |
| PDF | ButtonGroup | Range Slider | Breadcrumb |
| Word | Dropdown Menu | Toggle Switch Button | AppBar |
| PowerPoint | Progress Button | Signature | NOTIFICATION |
| LAYOUT | SplitButton | Rating | Toast |
| Dialog | Chips | FORMS | Progress Bar |
| ListView | Floating Action Button | In-place Editor | Spinner |
| Tooltip | Speed Dial | Query Builder | Badge |
| Splitter | | | Message |
| Dashboard | | | Skeleton |
| Card | | | |
| Avatar | | | |

<!-- markdownlint-enable MD033 MD018 -->

Getting Help

- If you are still not able to find the information that you are looking for in the self-help resources mentioned above then contact us by creating a support ticket in [our support site](#) or ask your query in Stack Overflow with the tag `syncfusion-ej2`.
- Don't see what you need? request it in our [feedback portal](#).

Note: Syncfusion does not collect any kind of information when our components are used in customer applications.

See also

- [Product Development](#)

- [Product Development Life Cycle](#)
- Syncfusion Project Template
- [Getting started with Syncfusion ASP.NET MVC Controls](#)

System Requirements for ASP.NET MVC EJ2 Components

To get start with ASP.NET MVC application, need to ensure the following software to be installed on the machine.

Integrated Development Environment (IDE)

ASP.NET MVC Applications can be developed using one of the following IDEs. You can also develop using [.NET CLI](#) without below IDEs.

- [Visual Studio 2022](#)
- [Visual Studio 2019](#)
- [Visual Studio 2017](#)
- [Visual Studio Code](#)

Frameworks

One of the the following .NET SDK is required to develop and run the Syncfusion UI controls for ASP.NET MVC.

- [.NET 7.0 SDK](#)
- [.NET 6.0 SDK](#)
- [.NET Core SDK 3.1](#)
- [.NET Core SDK 2.0](#)
- [.NET 4.5 Framework](#)

Browser Compatibility in ASP.NET MVC

The Syncfusion ASP.NET MVC controls are supported by all modern web browsers on both desktop and mobile devices running Windows, Linux, or MacOS.

| Browser | Versions |
|--|----------------|
| ----- | ----- |
| Google Chrome, including Android & iOS | Latest Version |
| Mozilla Firefox | Latest Version |
| Microsoft Edge | Latest Version |
| Apple Safari, including iOS | Latest Version |
| Opera | Latest Version |
| Microsoft Internet Explorer | 11 |

See also

- [ASP.NET MVC supported platforms](#)

NuGet Packages for ASP.NET MVC UI controls

NuGet is a Package management system for Visual Studio. It makes it easy to add, update and remove external libraries in our application. Syncfusion publishing all ASP.NET MVC NuGet packages in nuget.org. The Syncfusion ASP.NET MVC NuGet packages can be used without installing the Essential Studio or ASP.NET MVC platform installation to implement the Syncfusion ASP.NET MVC controls.

Available NuGet packages

Syncfusion.EJ2.MVC5

Syncfusion ASP.NET MVC UI controls are built from the ground up to be lightweight, responsive, and touch- friendly. It includes over 70+ controls, including grid, charts, scheduler, spreadsheet, gantt chart, listbox, file manager, rich text editor and much more.

<!-- markdownlint-disable MD033 -->

| NuGet package name | Controls | Dependencies |
|-------------------------------------|--------------|--|
| Syncfusion.EJ2.MVC5 | 70+ Controls | <ul style="list-style-type: none">• Newtonsoft.Json• Syncfusion.EJ2.JavaScript• Syncfusion.Licensing |

Syncfusion.EJ2.WordEditor.AspNet.Mvc5

The ASP.NET MVC Word Processor is a component with editing capabilities like Microsoft Word. Also known as the document editor, it is used to create, edit, view, and print Word documents. It provides all the common Word processing features including editing text, formatting contents, resizing images and tables, finding and replacing text, bookmarks, tables of contents, printing, and importing and exporting Word documents.

| NuGet package name | Controls | Dependencies |
|---|-----------------|--|
| Syncfusion.EJ2.WordEditor.AspNet.Mvc5 | Document Editor | <ul style="list-style-type: none">• Syncfusion.Compression.Base• Syncfusion.DocIO.AspNet.Mvc5• Syncfusion.OfficeChart.Base |

Syncfusion.EJ2.PdfViewer.AspNet.Mvc5

The ASP.NET MVC PDF Viewer supports viewing and reviewing PDF files in web applications and also printing them. The thumbnail, bookmark, hyperlink, and table of contents support provides easy navigation within and outside PDF files. The form-filling support provides a platform to fill and print with AcroForms. The PDF files can be reviewed with the available annotation tools.

| NuGet package name | Controls | Dependencies |
|--|------------|--|
| Syncfusion.EJ2.PdfViewer.AspNet.Mvc5 | PDF Viewer | <ul style="list-style-type: none">• Syncfusion.Compression.Base• Syncfusion.Pdf.AspNet.Mvc5 |

[Syncfusion.EJ2.SpellChecker.AspNet.Mvc5](#)

The ASP.NET MVC Word Processor is a component with editing capabilities like Microsoft Word. Also known as the document editor, it is used to create, edit, view, and print Word documents. It provides all the common Word processing features including editing text, formatting contents, resizing images and tables, finding and replacing text, bookmarks, tables of contents, printing, and importing and exporting Word documents.

| NuGet package name | Controls | Dependencies |
|---|-----------------|---|
| Syncfusion.EJ2.SpellChecker.AspNet.Mvc5 | Document Editor | <ul style="list-style-type: none"> This package has no dependencies. |

[Syncfusion.EJ2.Spreadsheet.AspNet.MVC5](#)

The ASP.NET Mvc Spreadsheet is also known as the ASP.NET Mvc Excel viewer, is a feature-rich control for organizing and analyzing data in tabular format. It provides all the common Excel features, including data binding, selection, editing, formatting, resizing, sorting, importing, and exporting Excel documents.

| NuGet package name | Controls | Dependencies |
|--|-------------|---|
| Syncfusion.EJ2.Spreadsheet.AspNet.MVC5 | Spreadsheet | <ul style="list-style-type: none"> This package has no dependencies. |

[Syncfusion.EJ2.GridExport.MVC5](#)

The ASP.NET Mvc Grid Export is an ideal library for visualizing the Grid data in a Excel, Pdf and CSV formats which also handles the data actions like filtering, sorting, Groupin etc.

| NuGet package name | Controls | Dependencies |
|--|-------------|---|
| Syncfusion.EJ2.GridExport.MVC5 | Grid Export | <ul style="list-style-type: none"> This package has no dependencies. |

Getting Started

[Getting Started with ASP.NET MVC using HTML Helper](#)

This article provides step-by-step instructions for building ASP.NET MVC application with Calendar control using HTML helper in Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for

[Syncfusion.EJ2.MVC5](#) and then install it. Alternatively, you can utilize the following package manager command to achieve the same.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

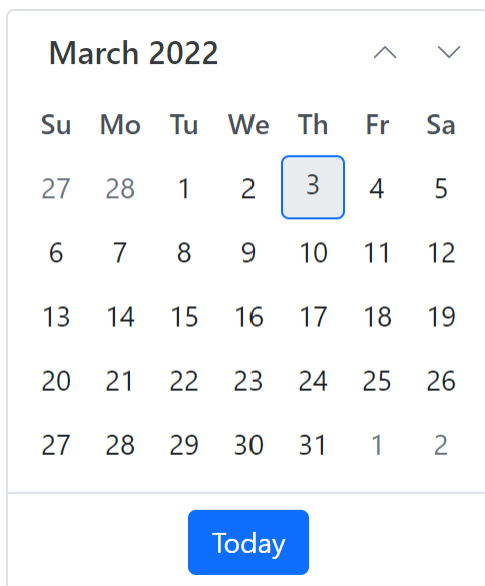
Add ASP.NET MVC Calendar control

Now, add the Syncfusion ASP.NET MVC Calendar control in `~/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().Calendar("calendar").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Calendar control will be rendered in the default web browser.



Getting Started with ASP.NET Core MVC using HTML Helper

This article provides step-by-step instructions for building ASP.NET Core MVC application with Calendar control using HTML helper in Visual Studio.

Prerequisites

[System requirements for ASP.NET Core components](#)

Create ASP.NET Core MVC application

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET Core Extension](#)

Install ASP.NET Core package in the application

To add **ASP.NET Core** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.AspNet.Core](#) and then install it. Alternatively, you can utilize the following package manager command to achieve the same.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.AspNet.Core -Version {{ site.releaseversion }}}
```

Note: Syncfusion ASP.NET Core controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.AspNet.Core NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Add Syncfusion ASP.NET Core namespace

Open `~/Views/_ViewImports.cshtml` file and add the `Syncfusion.EJ2` namespace.

~/ VIEWIMPORTS.CSHTML

```
@using WebApplication1
@using WebApplication1.Models
@using Syncfusion.EJ2
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Views/Shared/_Layout.cshtml` file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET Core controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/material.css" />
<!-- Syncfusion ASP.NET Core controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Views/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Note: Checkout the [Themes topic](#) to learn different ways ([CDN](#), [NPM package](#), and [CRG](#)) to refer styles in ASP.NET Core application, and to have the expected appearance for Syncfusion ASP.NET Core controls.

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET Core application.

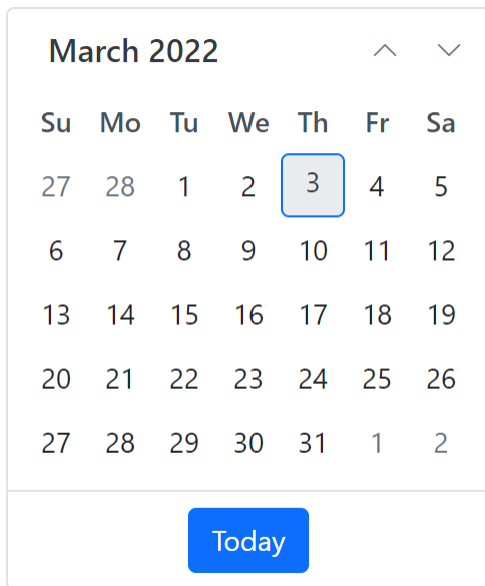
Add ASP.NET Core Calendar control

Now, add the Syncfusion ASP.NET Core Calendar Html helper in `~/Views/Home/Index.cshtml` page.

CSHTML

```
<div>
@Html.EJS().Calendar("calendar").Render()
</div>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET Core Calendar control will be rendered in the default web browser.



See also

- [Getting Started with Syncfusion ASP.NET Core using Razor Pages](#)
- [Getting Started with Syncfusion ASP.NET Core MVC using Tag Helper](#)

Getting Started with ASP.NET Core MVC using Tag Helper

This article provides step-by-step instructions for building ASP.NET Core MVC application with Calendar control using tag helper in Visual Studio.

Prerequisites

[System requirements for ASP.NET Core components](#)

Create ASP.NET Core MVC application

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET Core Extension](#)

Install ASP.NET Core package in the application

To add **ASP.NET Core** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for

[Syncfusion.EJ2.AspNet.Core](#) and then install it. Alternatively, you can utilize the following package manager command to achieve the same.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.AspNet.Core -Version {{ site.releaseversion }}}
```

Note: Syncfusion ASP.NET Core controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.AspNet.Core NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Add Syncfusion ASP.NET Core Tag Helper

Open `~/Views/_ViewImports.cshtml` file and import the `Syncfusion.EJ2` TagHelper.

~/ VIEWIMPORTS.CSHTML

```
@addTagHelper *, Syncfusion.EJ2
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Views/Shared/_Layout.cshtml` file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET Core controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/material.css" />
<!-- Syncfusion ASP.NET Core controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Also, register the script manager `<ejs-script>` at the end of `<body>` in the ASP.NET Core application as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET Core Script Manager -->
<ejs-scripts></ejs-scripts>
</body>
```

Note: Checkout the [Themes topic](#) to learn different ways ([CDN](#), [NPM package](#), and [CRG](#)) to refer styles in ASP.NET Core application, and to have the expected appearance for Syncfusion ASP.NET Core controls.

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET Core application.

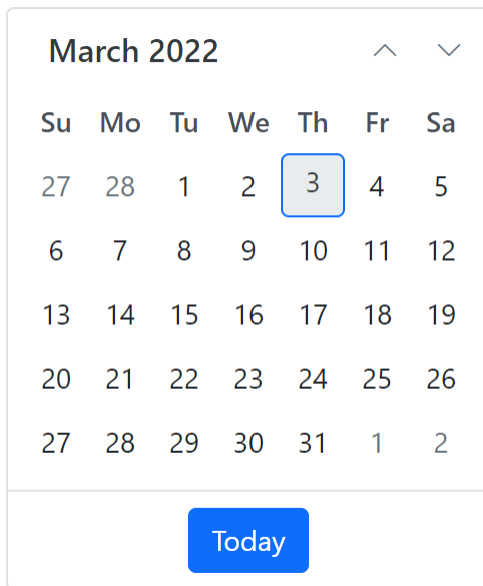
Add ASP.NET Core Calendar control

Now, add the Syncfusion ASP.NET Core Calendar tag helper in `~/Views/Home/Index.cshtml` page.

CSHTML

```
<div>
<ejs-calendar id="calendar"></ejs-calendar>
</div>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET Core Calendar control will be rendered in the default web browser.



See also

- [Getting Started with Syncfusion ASP.NET Core using Razor Pages](#)
- [Getting Started with Syncfusion ASP.NET Core MVC using HTML Helper](#)

Visual Studio extensions

Create project

Syncfusion provides the **Visual Studio Project Templates** for the Syncfusion ASP.NET MVC platform to create the Syncfusion ASP.NET MVC Web Application with the Essential JS 2 components.

Note: The Syncfusion ASP.NET MVC (Essential JS 2) project templates are available from v16.2.0.41.

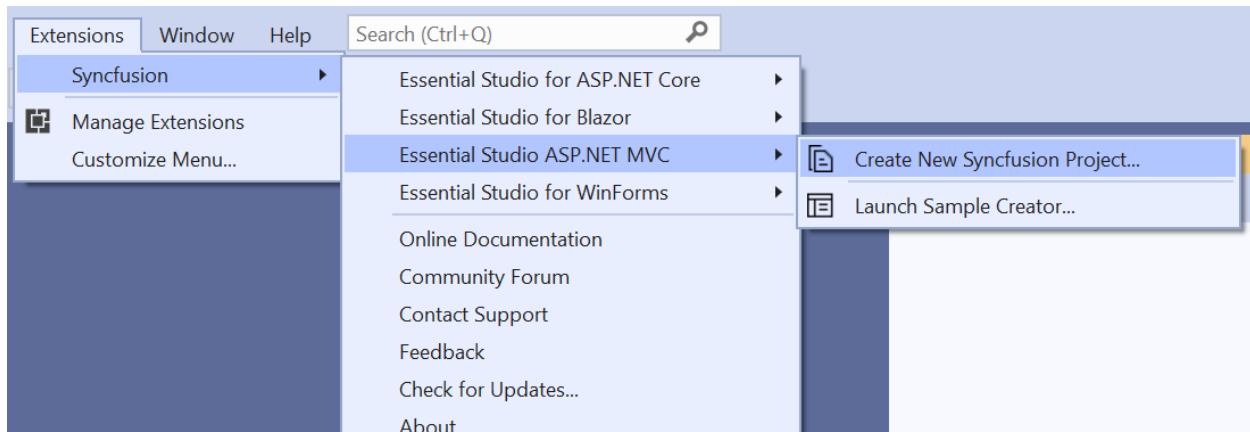
Use the following steps to create the **Syncfusion ASP.NET MVC (Essential JS 2) Web Application** through the **Visual Studio Project Template**.

Note: Before use the Syncfusion ASP.NET MVC Project Template, check whether the **ASP.NET MVC Extensions - Syncfusion** installed or not in Visual Studio Extension Manager by clicking on the **Extensions -> Manage Extensions -> Installed** for Visual Studio 2019 or later and for Visual Studio 2017 or lower by clicking on the **Tools -> Extensions and Updates -> Installed**. If this extension not installed, install the extension by follow the steps from the [download and installation](#) help topic.

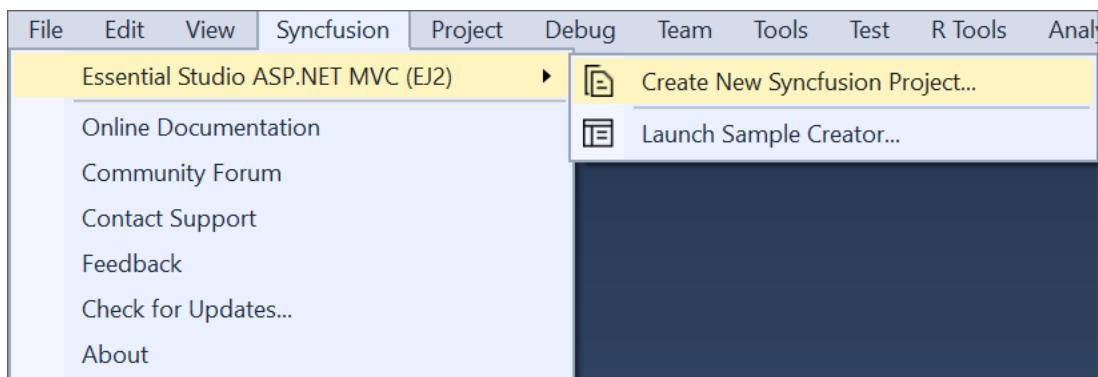
1. To create the Syncfusion ASP.NET MVC (Essential JS 2) project, follow either one of the options below:

Option 1:

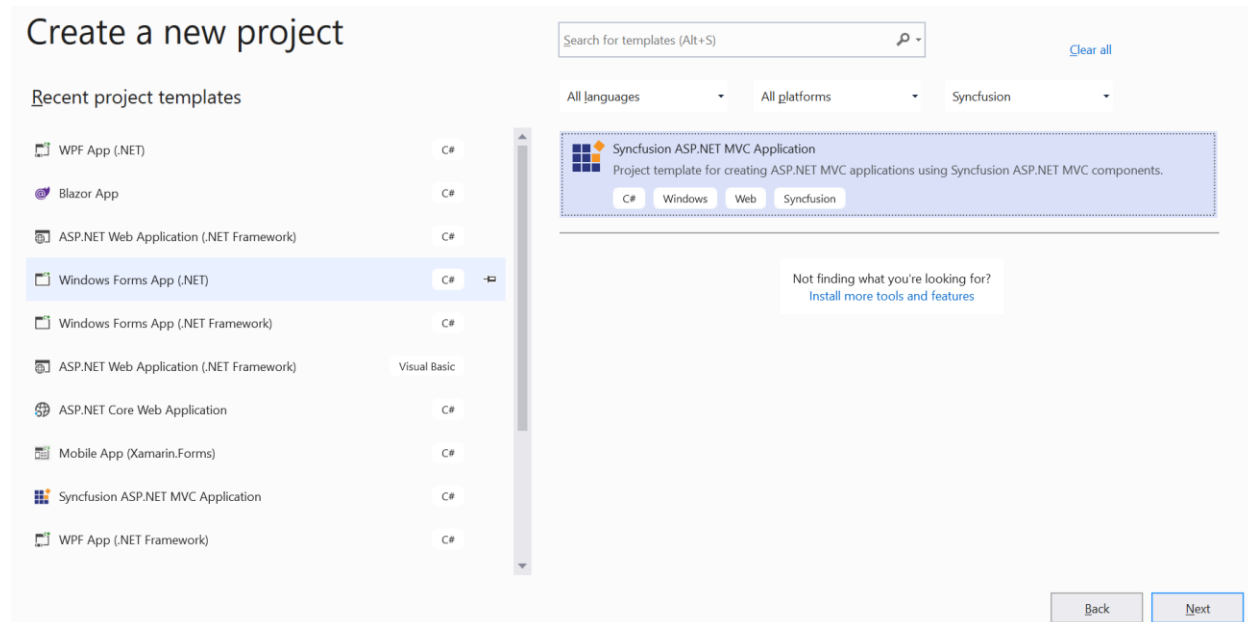
Click **Extensions->Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ2) > Create New Syncfusion Project...** in Visual Studio menu.



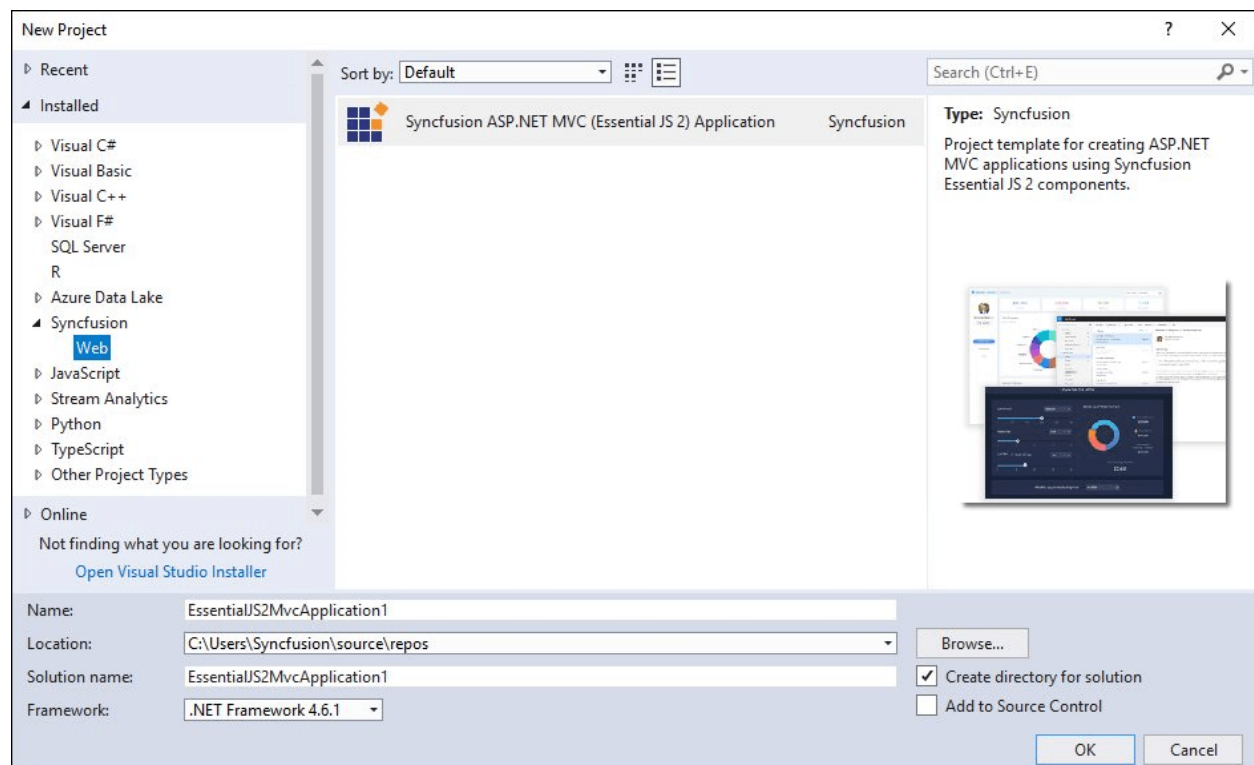
Note: In Visual Studio 2017 or lower, you can see the Syncfusion menu directly in the Visual Studio menu.

**Option 2:**

Choose **File -> New -> Project**. Opens a new dialog to create a new project. By filtering the project type with Syncfusion or using the **Syncfusion** keyword in the search option, you can get the templates offered by Syncfusion for ASP.NET MVC.



Note: In Visual Studio 2017 or lower, Choose **File > New > Project** and navigate to **Syncfusion > Web > Syncfusion ASP.NET MVC (Essential JS 2) Application**.



2. Name the **Project**, choose the **destination location**, and set the .NET Framework of the project, and then click **OK**. The Project Configuration Wizard appears.

ASP.NET MVC project configuration wizard

Version 19.4.0.38

×

Project Preference

Target MVC Version

MVC5

Themes

Bootstrap v5

Assets From

NuGet

Theme Preview

Preview for your selected theme is shown as follows:

Button

Primary

Grid

PRODUCT ID	PRODUCT NAME	UNIT PRICE	UNITS IN STOCK
1	Chai	\$18.00	39
2	Chang	\$19.00	17
3	Aniseed Syrup	\$10.00	13

<<

<

1

2

3

4

5

...

>

>>

1 to 5 of 77

Syncfusion

Help

Support

Suggest a Feature

What's New

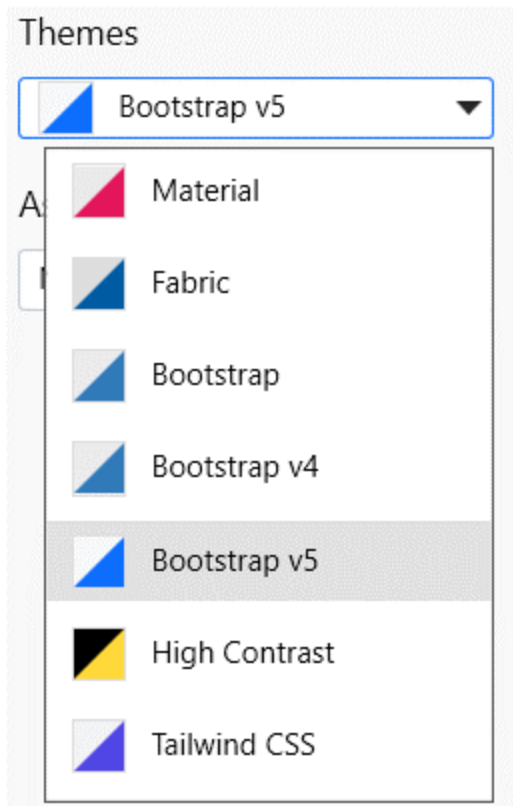
Create

Cancel

Project configurations

Target MVC Version: Select the version of ASP.NET MVC Project, either MVC5 or MVC4.

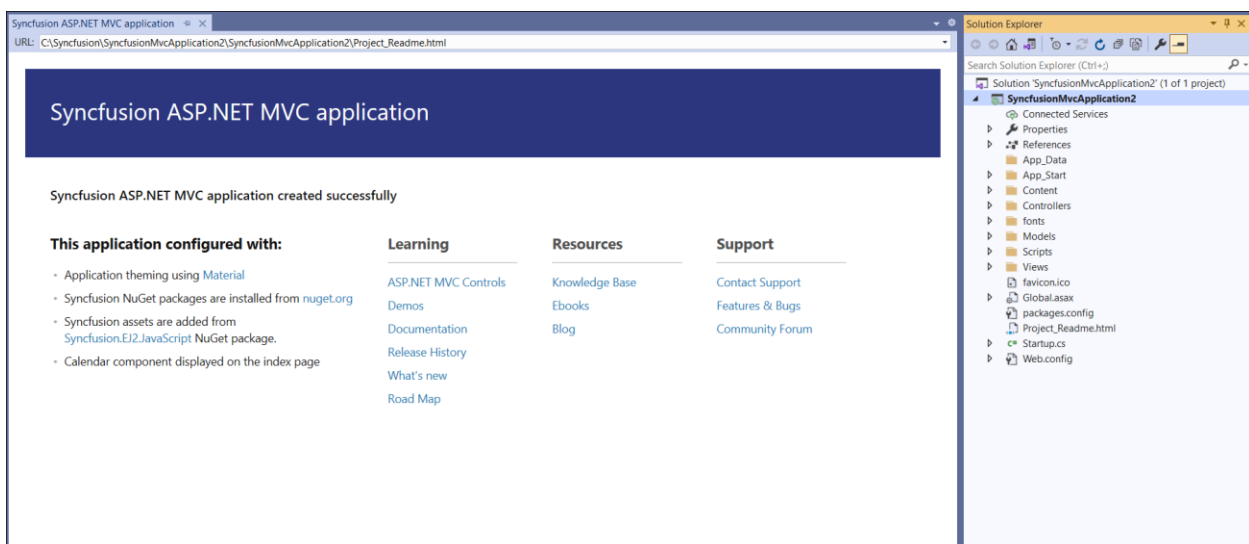
Theme Selection: Choose the required Theme. The Theme Preview section shows the controls preview with selected theme before creating the Syncfusion project.



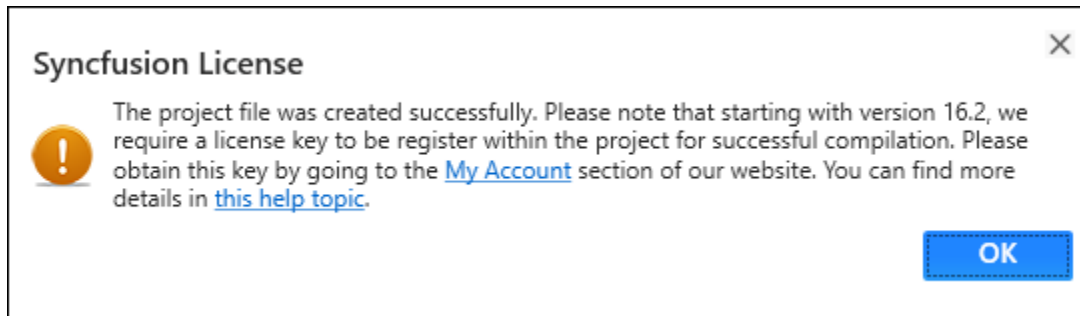
Assets From: : Load the Syncfusion Essential JS 2 assets to ASP.NET MVC Project, either NuGet, CDN, or Installed Location.

Note: Installed location option will be available only when the Syncfusion Essential JavaScript 2 setup has been installed.

3. Click **Create**, the Syncfusion ASP.NET MVC (Essential JS 2) Application will be created.



4. The created Syncfusion ASP.NET MVC application configures with most recent Syncfusion ASP.NET MVC NuGet packages, selected style and scripts for use Syncfusion components.
5. Then, the Syncfusion licensing registration required message box will be shown, if you installed the trial setup or NuGet packages since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Navigate to the [help topic](#), which is shown in the licensing message box to generate and register the Syncfusion license key to your project. Refer to this [blog](#) post to learn more about the licensing changes introduced in Essential Studio.



Installation

Web Installer

Downloading Syncfusion ASP.NET MVC EJ2 web installer

The Syncfusion ASP . NET MVC - EJ2 Installer can be downloaded from the [Syncfusion](#) website. You can either download the licensed installer or try our trial installer depending on your license.

- Trial Installer
- Licensed Installer

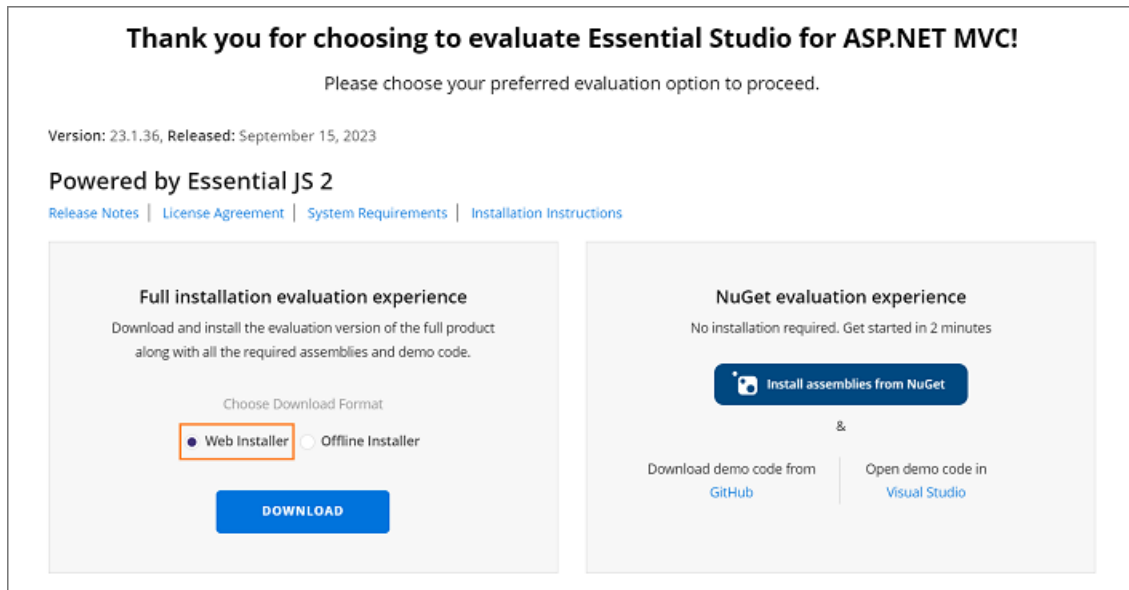
[Download the Trial Version](#)

Our 30-day trial can be downloaded in two ways.

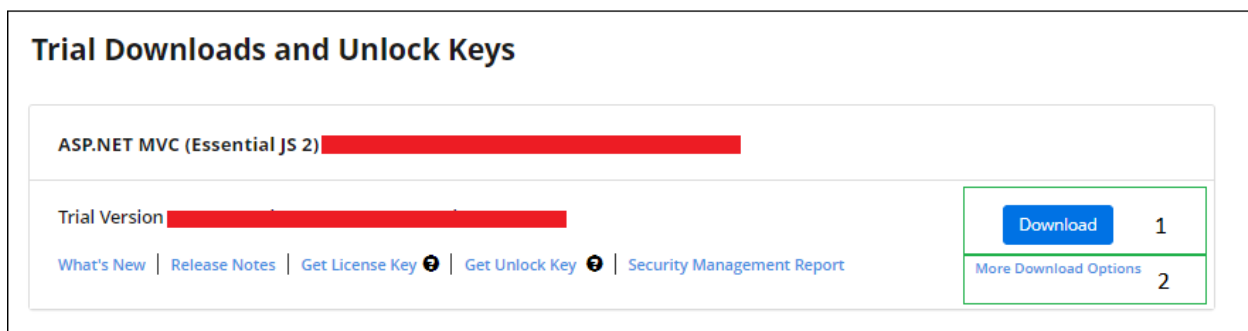
- Download Free Trial Setup
- Start Trials if using components through [NuGet.org](#)

[Download Free Trial Setup](#)

1. You can evaluate our 30-day free trial by visiting the [Download Free Trial](#) page and select the ASP . NET MVC platform.
2. After completing the required form or logging in with your registered Syncfusion account, you can download the ASP . NET MVC EJ2 trial installer from the confirmation page. (See the screenshot below)



3. With a trial license, only the latest version's trial installer can be downloaded.
4. After downloading, the Syncfusion ASP . NET MVC - EJ2 trial installer can be unlocked using either the trial unlock key or the Syncfusion registered login credential. More information on generating an unlock key can be found in [this](#) article.
5. Before the trial expires, you can download the trial installer at any time from your registered account's [Trials & Downloads](#) page. (See the screenshot below)
6. Click the Download (element 1 in the screenshot below) button to get the Syncfusion Essential Studio ASP . NET MVC - EJ2 web installer.



Start Trials if using components through [NuGet.org](#)

You should initiate an evaluation if you have already obtained our components through [NuGet.org](https://www.nuget.org/packages?q=syncfusion)

1. You can start your 30-day free trial for ASP . NET MVC - EJ2 from the [Start Trial](#) page from your account.



2. To access this page, you must sign up/ log in with your Syncfusion account.
3. Begin your trial by selecting the ASP . NET MVC - EJ2 product.

Note: If you've already used the trial products and they haven't expired, you won't be able to start the trial for the same product again.

4. After you've started the trial, go to the [Trials & Downloads](#) page to get the latest version trial installer. You can generate the [unlock key](#) and [license key](#) here at any time before the trial period expires. (See the screenshot below)



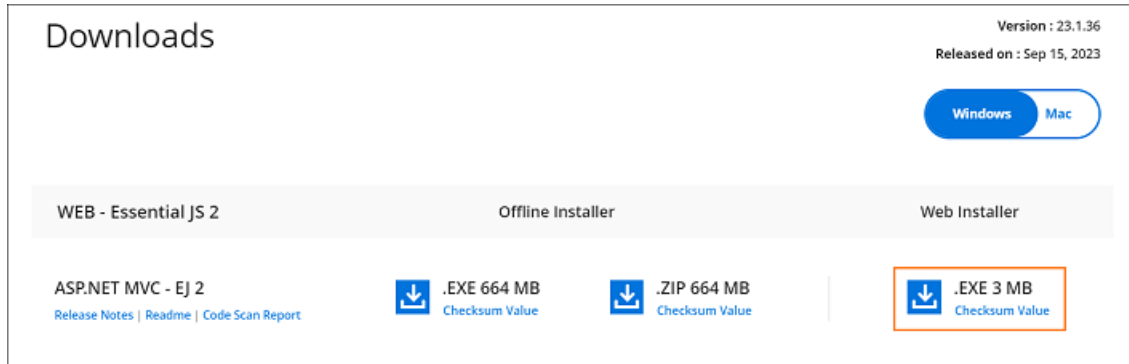
5. You can find your current active trial products on the [Trials & Downloads](#) page.

Download the License Version

1. Syncfusion licensed products will be available in the [License & Downloads](#) page under your registered Syncfusion account.
2. You can view all the licenses (both active and expired) associated with your account.
3. Click the Download (element 1 in the screenshot below) button to download the respective product's installer.
4. The most recent version of the installer will be downloaded from this page.
5. To download older version installers, go to [Downloads Older Versions](#) (element 2 in the screenshot below).
6. You can download other platform / add-on installers by going to More Downloads Options (element 3 in the screenshot below).



7. Before the license expires, you can download the installer at any time from your registered account's [License & Downloads](#) page (See the screenshot below.)



8. After downloading, the Syncfusion ASP.NET MVC EJ2 web installer can be unlocked using Syncfusion registered login credential.

Note: For Syncfusion trial and licensed products, there is no separate web installer. Based on your account license, Syncfusion trial or licensed products will be installed via web installer.

You can also refer to the [Web installer](#) links for step-by-step installation guidelines.

Installing Syncfusion ASP.NET MVC EJ2 Web Installer

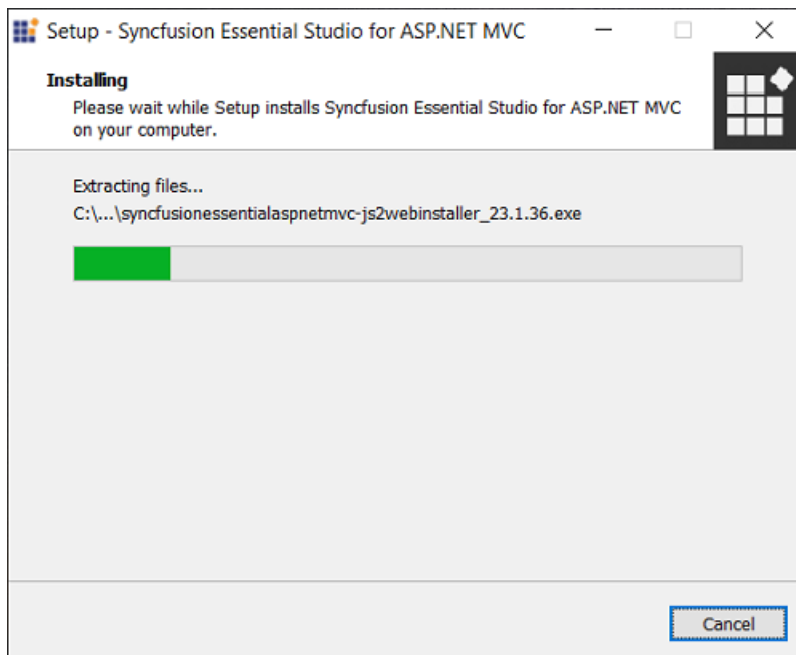
Overview

For the Essential Studio ASP . NET MVC - EJ2 product, Syncfusion offers a Web Installer. This installer alleviates the burden of downloading a larger installer. You can simply download and run the online installer, which will be smaller in size and will download and install the Essential Studio products you have chosen. You can get the most recent version of Essential Studio Web Installer [here](#).

Installation

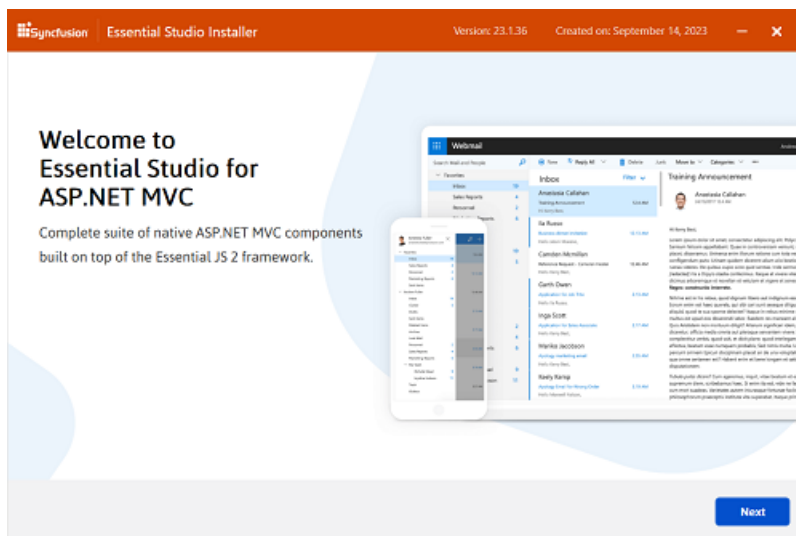
The steps below show how to install Essential Studio ASP . NET MVC - EJ2 Web Installer.

1. Open the Syncfusion Essential Studio ASP . NET MVC - EJ2 Web Installer file from downloaded location by double-clicking it. The Installer Wizard automatically open and extracts the package.



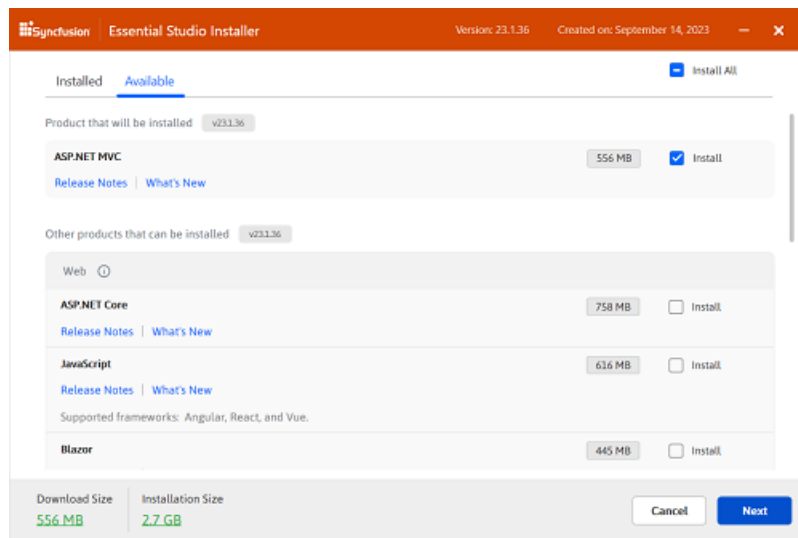
Note: The installer wizard extracts the `syncfusionessentialaspnetmvc-js2webinstaller_{version}.exe` dialog, which displays the package's unzip operation.

2. The Syncfusion ASP . NET MVC - EJ2 Web Installer's welcome wizard will be displayed. Click the Next button.



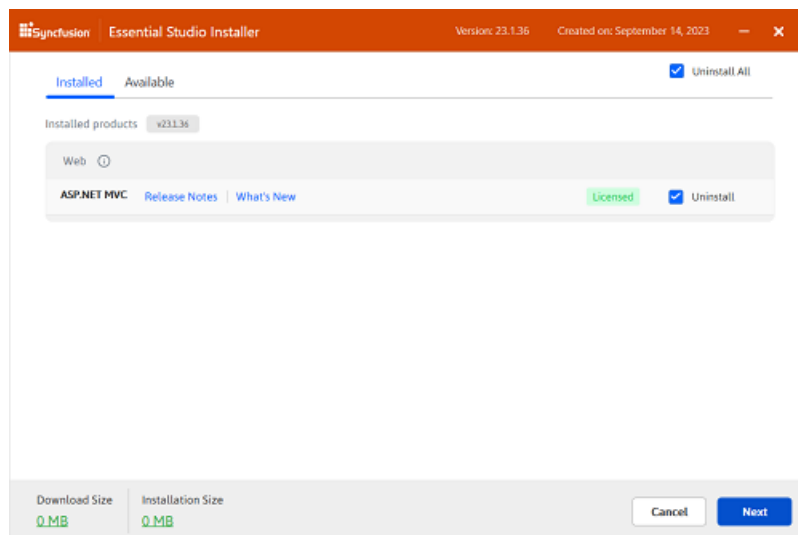
3. The Platform Selection Wizard will appear. From the **Available** tab, select the products to be installed. Select the **Install All** checkbox to install all products.

Available



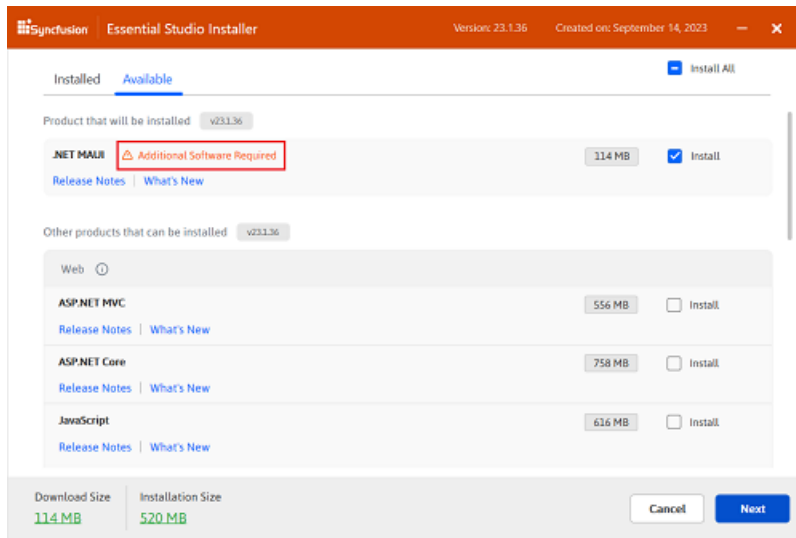
If you have multiple products installed in the same version, they will be listed under the **Installed** tab. You can also select which products to uninstall from the same version. Click the Next button.

Installed

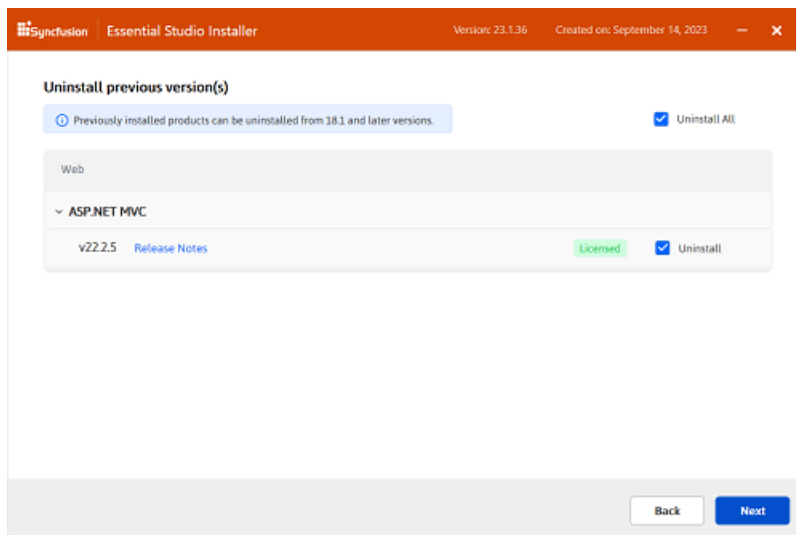


Information: If the required software for the selected product isn't already installed, the **Additional Software Required** alert will be appear. You can, however, continue the installation and install the necessary software later.

Required Software

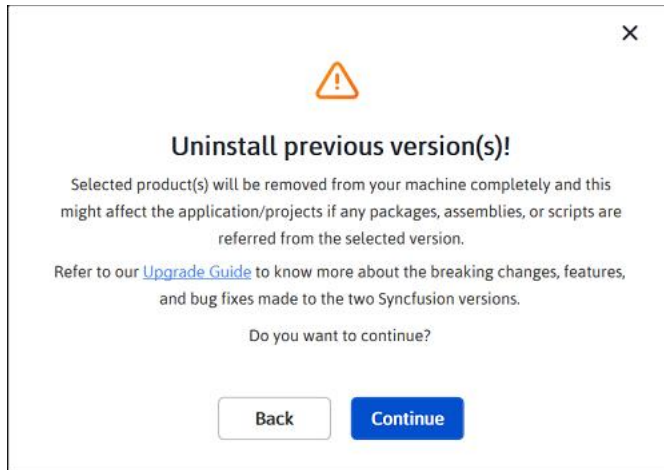


4. If previous version(s) for the selected products are installed, the Uninstall previous version wizard will be displayed. You can see the list of previously installed versions for the products you've chosen here. To remove all versions, check the **Uninstall All** checkbox. Click the Next button.

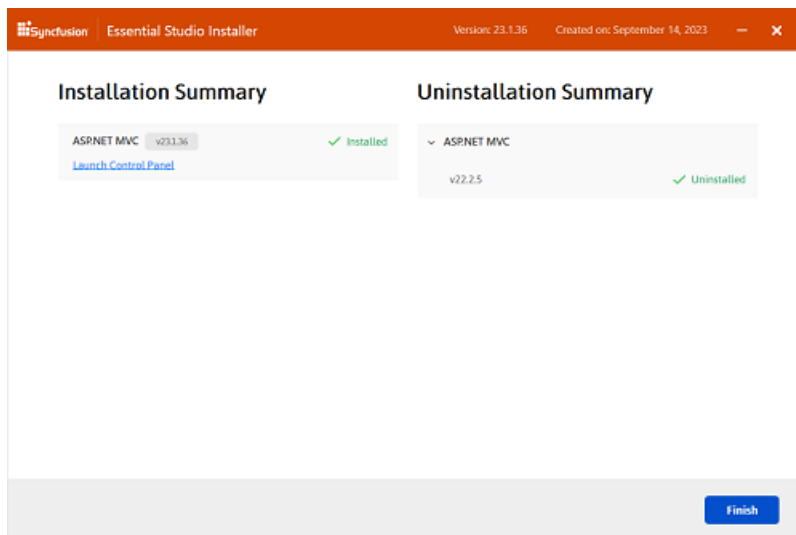


Note: From the 2021 Volume 1 release, Syncfusion has provided option to uninstall the previous versions from 18.1 while installing the new version.

5. Pop up screen will be displayed to get the confirmation to uninstall selected previous versions.



6. The Confirmation Wizard will appear with the list of products to be installed/uninstalled. You can view and modify the list of products that will be installed and uninstalled from this page.



Note: By clicking the **Download Size and Installation Size** links, you can determine the approximate size of the download and installation

7. The Configuration Wizard will appear. You can change the Download, Install, and Demos locations from here. You can also change the Additional settings on a product-by-product basis. Click Next to install with the default settings.

Configuration

Download Location
C:\ProgramData\Syncfusion\23.1.36\Downloads\ [Browse](#)

Installation Location
C:\Program Files (x86)\Syncfusion\Essential Studio\ [Browse](#)

Demos Location
C:\Users\Public\Documents\Syncfusion\ [Browse](#)

☒ I Agree to the [License Terms](#) and [Privacy Policy](#)

Additional Settings

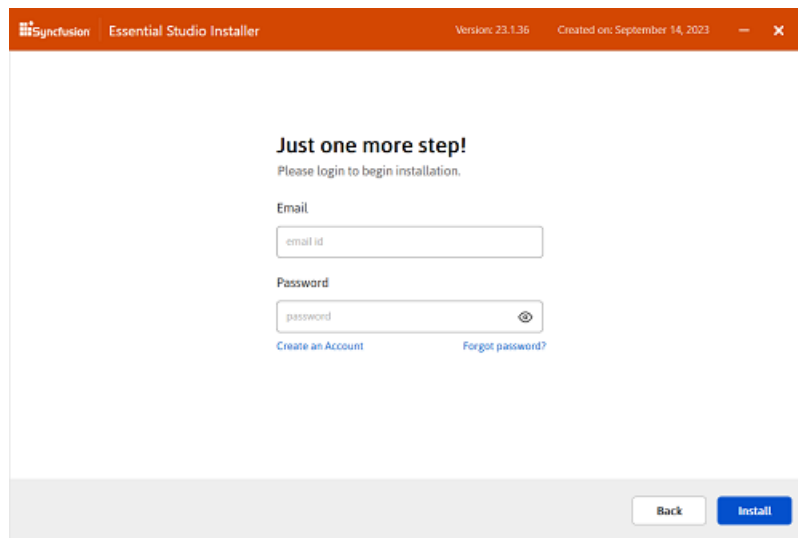
- ☒ Install Demos
 - ☒ ASP.NET MVC
- ☒ Configure Syncfusion Extensions in Visual Studio
 - ☒ ASP.NET MVC
- ☒ Create Desktop Shortcut(s)
 - ☒ ASP.NET MVC
- ☒ Create Start Menu Shortcut(s)
 - ☒ ASP.NET MVC

Download Size: 556 MB | Installation Size: 2.7 GB

[Back](#) [Next](#)

Additional Settings

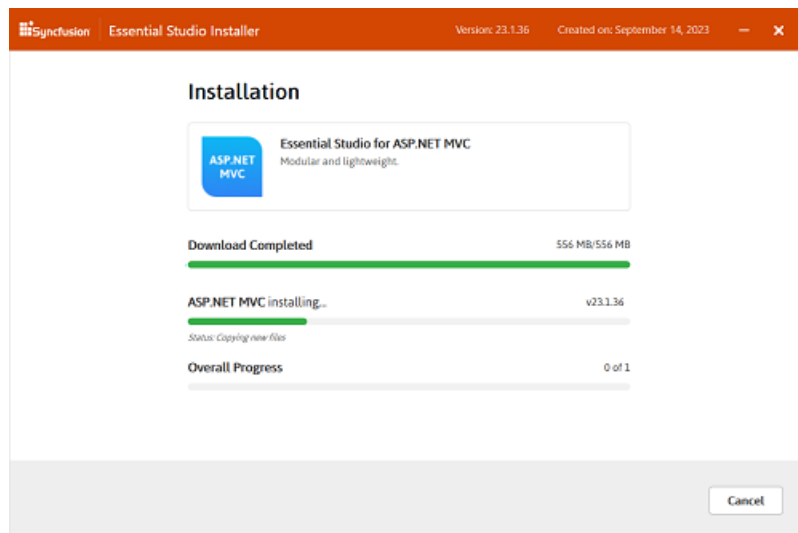
- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box unchecked, if you do not want to install Syncfusion samples.
- Select the **Configure Syncfusion Extensions controls in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.
- Check the **Create Desktop Shortcut** checkbox to add a desktop shortcut for Syncfusion Control Panel.
- Check the **Create Start Menu Shortcut** checkbox to add a shortcut to the start menu for Syncfusion Control Panel.
 8. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click the Next button.
 9. The login wizard will appear. You must enter your Syncfusion email address and password. If you do not already have a Syncfusion account, you can create one by clicking on **Create an Account**. If you have forgotten your password, click **Forgot Password** to create a new one. Click the Install button.



The screenshot shows the 'Essential Studio Installer' window. The title bar includes the Syncfusion logo, the text 'Essential Studio Installer', and version/creation information: 'Version: 23.1.36 Created on: September 14, 2023'. The main content area has the heading 'Just one more step!' followed by the instruction 'Please login to begin installation.' Below this are two input fields: 'Email' (with placeholder 'email id') and 'Password' (with placeholder 'password' and a toggle icon). At the bottom of the password field are two links: 'Create an Account' and 'Forgot password?'. At the very bottom of the window are two buttons: 'Back' and 'Install'.

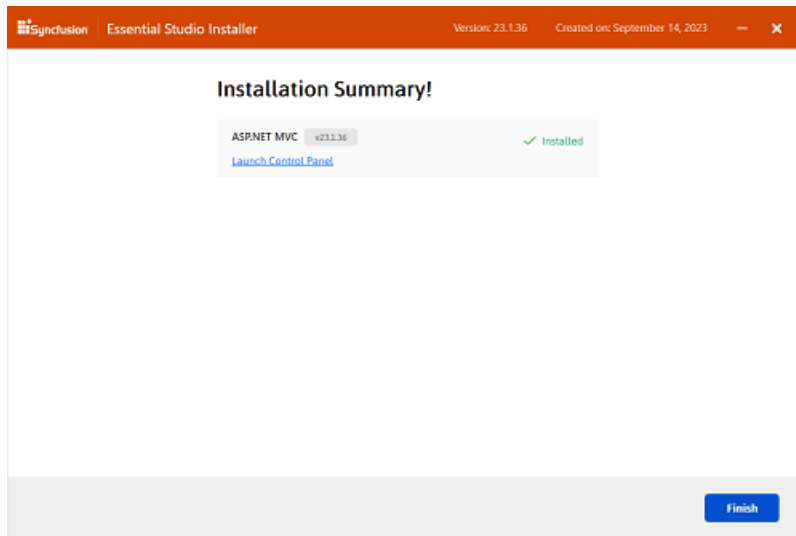
Information: The products you have chosen will be installed based on your Syncfusion License (Trial or Licensed).

10. The download and installation/uninstallation progress will be displayed as shown below.



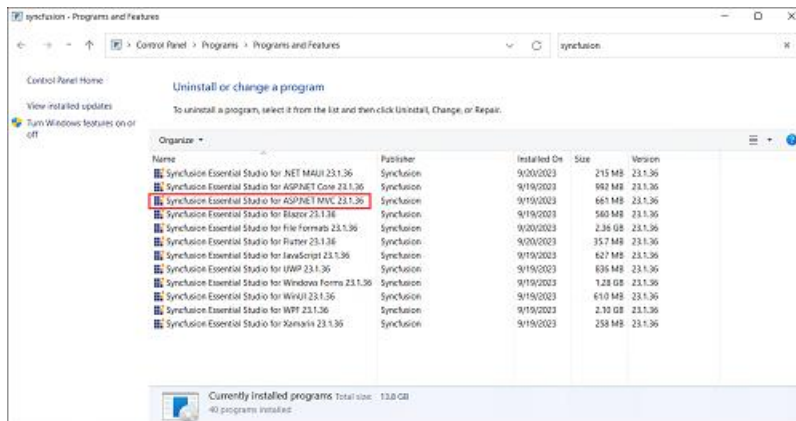
The screenshot shows the 'Installation' progress screen of the 'Essential Studio Installer'. The title bar is the same as the previous screen. The main content area features a box for 'Essential Studio for ASP.NET MVC' with the tagline 'Modular and lightweight.' Below this box are three progress indicators: 1. 'Download Completed' with a green bar at 100% and text '556 MB/556 MB'. 2. 'ASP.NET MVC installing..' with a green bar at approximately 50% and text 'v23.1.36'. Below this is the status 'Status: Copying new files'. 3. 'Overall Progress' with a grey bar at 0% and text '0 of 1'. At the bottom right of the window is a 'Cancel' button.

11. When the installation is finished, the **Summary** wizard will appear. Here you can see the list of products that have been installed successfully and those that have failed. To close the Summary wizard, click Finish.



To open the Syncfusion Control Panel, click **Launch Control Panel**

12. After installation, there will be two Syncfusion control panel entries, as shown below. The Essential Studio entry will manage all Syncfusion products installed in the same version, while the Product entry will only uninstall the specific product setup.



Uninstallation

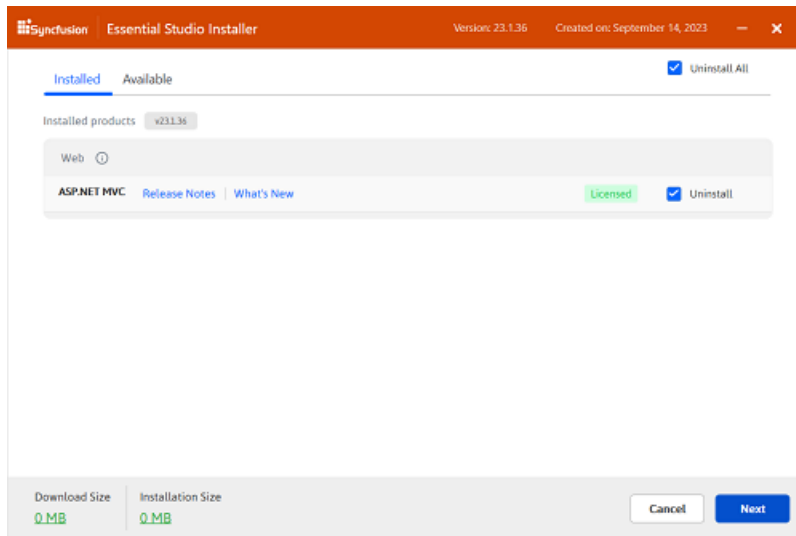
Syncfusion ASP . NET MVC - EJ2 installer can be uninstalled in two ways.

- Uninstall the ASP . NET MVC - EJ2 using the Syncfusion ASP . NET MVC - EJ2 web installer.
- Uninstall the ASP . NET MVC - EJ2 from Windows Control Panel.

Follow either one of the option below to uninstall Syncfusion Essential Studio ASP . NET MVC - EJ2 installer.

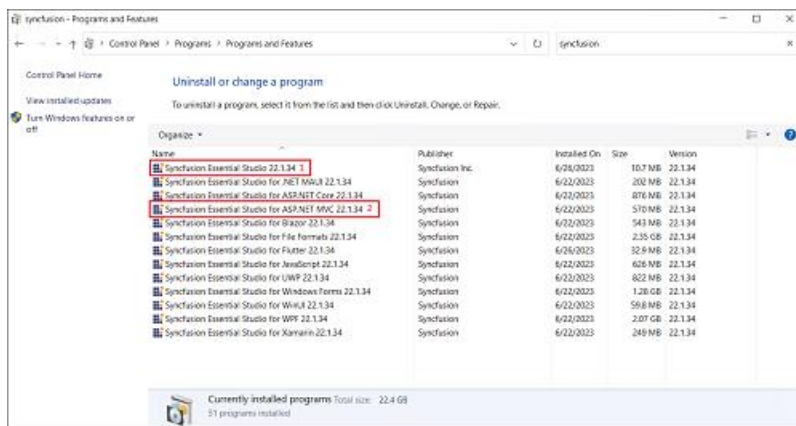
Option 1: Uninstall the ASP . NET MVC - EJ2 using the Syncfusion ASP . NET MVC - EJ2 web installer

Syncfusion provides the option to uninstall products of the same version directly from the Web Installer application. Select the products to be uninstalled from the list, and Web Installer will uninstall them one by one.



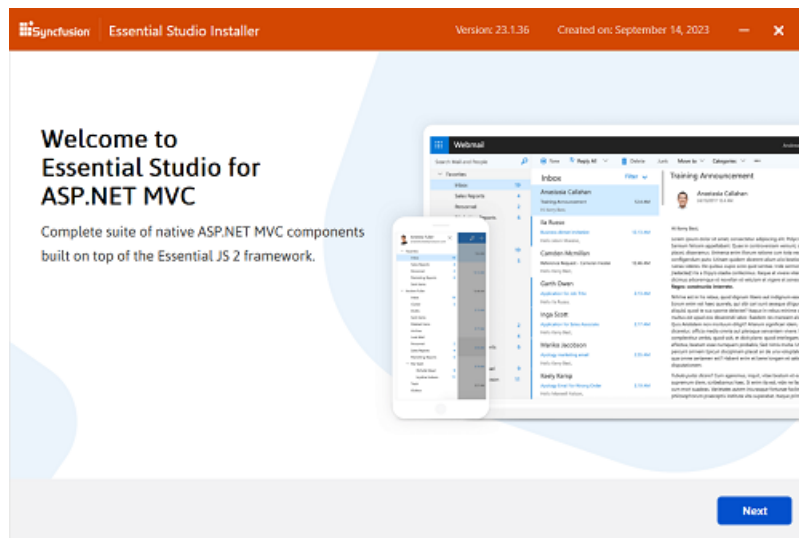
Option 2: Uninstall the ASP . NET MVC - EJ2 from Windows Control Panel

You can uninstall all the installed products by selecting the **Syncfusion Essential Studio {version}** entry (element 1 in the below screenshot) from the Windows control panel, or you can uninstall ASP . NET MVC - EJ2 alone by selecting the **Syncfusion Essential Studio for ASP . NET MVC {version}** entry (element 2 in the below screenshot) from the Windows control panel.



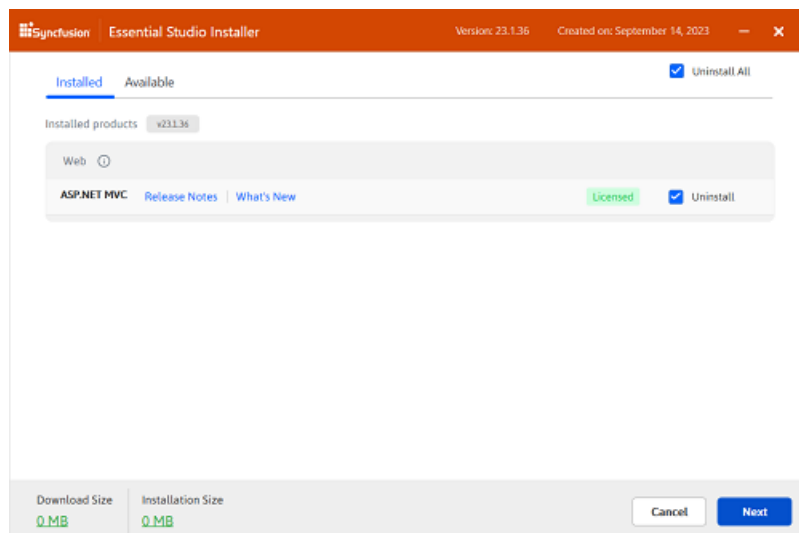
Note: If the **Syncfusion Essential Studio for ASP . NET MVC - EJ2 {version}** entry is selected from the Windows control panel, the Syncfusion Essential Studio ASP . NET MVC - EJ2 alone will be removed and the below default MSI uninstallation window will be displayed.

1. The Syncfusion ASP . NET MVC - EJ2 Web Installer's welcome wizard will be displayed. Click the Next button.



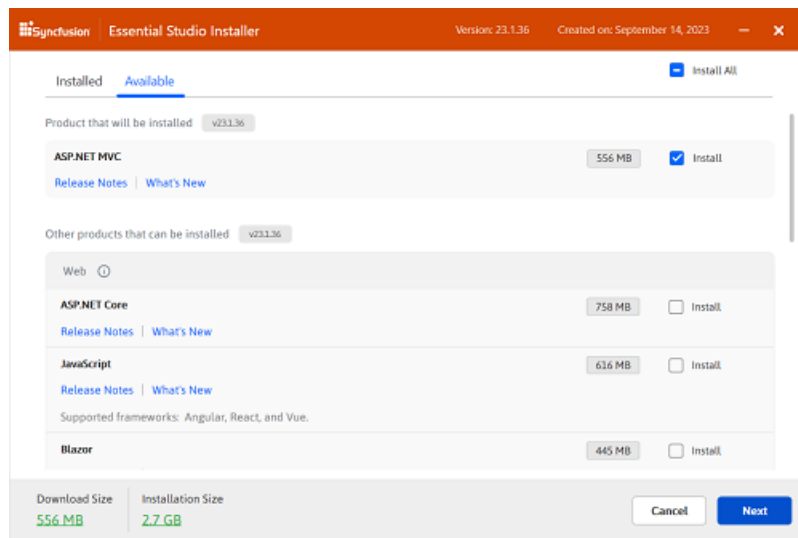
- The Platform Selection Wizard will appear. From the **Installed** tab, select the products to be uninstalled. To select all the products, check the **Uninstall All** checkbox. Click the Next button.

Installed

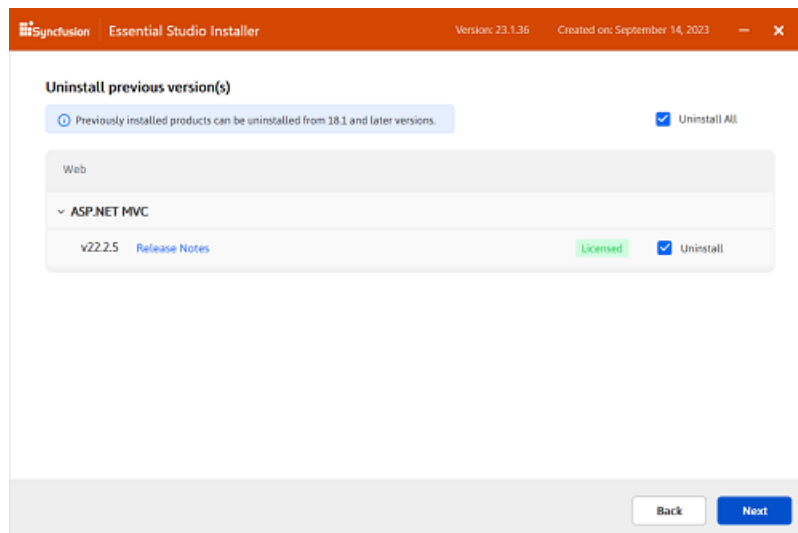


You can also select the products to be installed from the **Available** tab. Click the Next button.

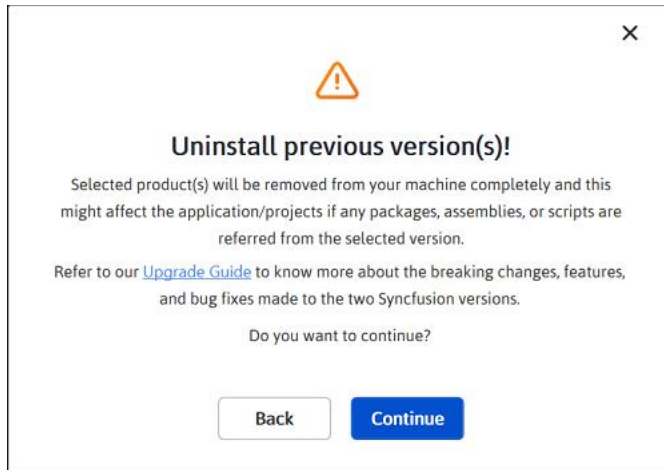
Available



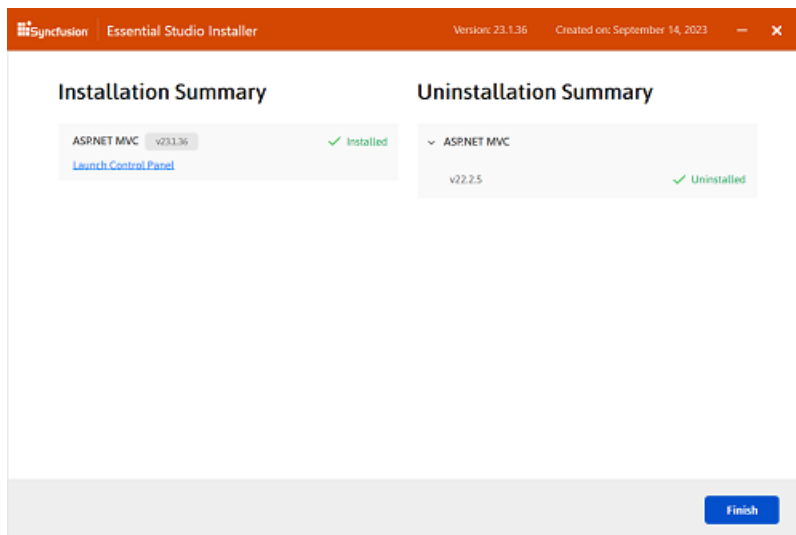
3. If any other products selected for installation, Uninstall previous version wizard will be displayed with the previous version(s) installed for the selected products. Here you can view the list of installed previous versions for the selected products. Select **Uninstall All** checkbox to select all the versions. Click Next.



4. Pop up screen will be displayed to get the confirmation to uninstall selected previous versions.



5. The Confirmation Wizard will appear with the list of products to be installed/uninstalled. Here you can view and modify the list of products that will be installed/uninstalled.



Note: By clicking the **Download Size and Installation Size** links, you can determine the approximate size of the download and installation

6. The Configuration Wizard will appear. You can change the Download, Install, and Demos locations from here. You can also change the Additional settings on a product-by-product basis. Click Next to install with the default settings.

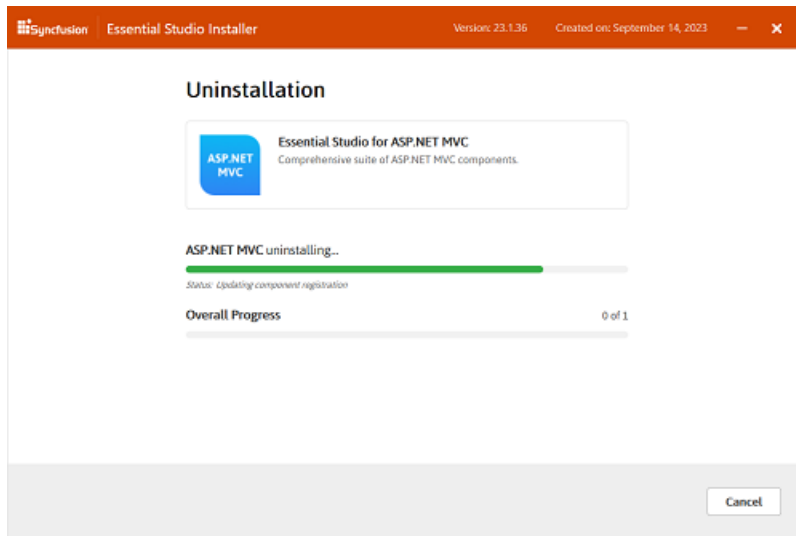
The screenshot shows the 'Configuration' window of the Syncfusion Essential Studio Installer. The window has an orange header bar with the Syncfusion logo, 'Essential Studio Installer', version '23.1.36', and creation date 'September 14, 2023'. The main content area is divided into two columns. The left column contains three sections: 'Download Location' with a text box showing 'C:\ProgramData\Syncfusion\23.1.36\Downloads\' and a 'Browse' button; 'Installation Location' with a text box showing 'C:\Program Files (x86)\Syncfusion\Essential Studio\' and a 'Browse' button; and 'Demos Location' with a text box showing 'C:\Users\Public\Documents\Syncfusion\' and a 'Browse' button. Below these is a checkbox labeled 'I Agree to the [License Terms](#) and [Privacy Policy](#)', which is checked. The right column is titled 'Additional Settings' and contains a list of checkboxes: 'Install Demos', 'ASP.NET MVC', 'Configure Syncfusion Extensions in Visual Studio', 'ASP.NET MVC', 'Create Desktop Shortcut(s)', 'ASP.NET MVC', 'Create Start Menu Shortcut(s)', and 'ASP.NET MVC'. All these checkboxes are checked. At the bottom left, there is a summary table: 'Download Size' is '556 MB' and 'Installation Size' is '2.7 GB'. At the bottom right are 'Back' and 'Next' buttons.

7. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click the Next button.
8. The login wizard will appear. You must enter your Syncfusion email address and password. If you do not already have a Syncfusion account, you can create one by clicking on **Create an Account**. If you have forgotten your password, click **Forgot Password** to create a new one. Click the Install button.

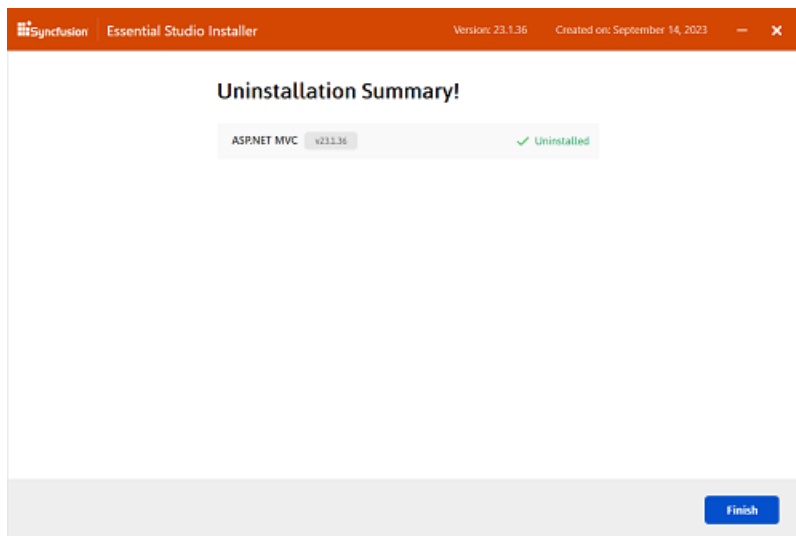
The screenshot shows the 'Just one more step!' login window of the Syncfusion Essential Studio Installer. The window has an orange header bar with the Syncfusion logo, 'Essential Studio Installer', version '23.1.36', and creation date 'September 14, 2023'. The main content area is white and contains the text 'Just one more step!' followed by 'Please login to begin installation.' Below this are two input fields: 'Email' with a placeholder 'email id' and 'Password' with a placeholder 'password' and a toggle icon. At the bottom left are links for 'Create an Account' and 'Forgot password?'. At the bottom right are 'Back' and 'Install' buttons.

Information: The products you have chosen will be installed based on your Syncfusion License(Trial or Licensed).

9. The download, installation and uninstallation progresses will be shown.



10. When the installation is finished, the **Summary** wizard will appear. Here you can see the list of products that have been successfully and unsuccessfully installed/uninstalled. To close the Summary wizard, click Finish.



- To open the Syncfusion Control Panel, click **Launch Control Panel**.

Offline Installer

Downloading Syncfusion ASP.NET MVC EJ2 offline installer

The Syncfusion ASP . NET MVC - EJ2 Installer can be downloaded from the [Syncfusion](https://www.syncfusion.com/) website. You can either download the licensed installer or try our trial installer depending on your license.

- Trial Installer
- Licensed Installer

Download the Trial Version

Our 30-day trial can be downloaded in two ways.

- Download Free Trial Setup
- Start Trials if using components through [NuGet.org](https://nuget.org)

Download Free Trial Setup

1. You can evaluate our 30-day free trial by visiting the [Download Free Trial](#) page and select the ASP . NET MVC platform.
2. After completing the required form or logging in with your registered Syncfusion account, you can download the ASP . NET MVC EJ2 trial installer from the confirmation page. (See the screenshot below)

Thank you for choosing to evaluate Essential Studio for ASP.NET MVC!

Please choose your preferred evaluation option to proceed.

Version: 23.1.36, Released: September 15, 2023

Powered by Essential JS 2

[Release Notes](#) | [License Agreement](#) | [System Requirements](#) | [Installation Instructions](#)

Full installation evaluation experience

Download and install the evaluation version of the full product along with all the required assemblies and demo code.

Choose Download Format

☐ Web Installer ☒ **Offline Installer**

DOWNLOAD **.EXE**

NuGet evaluation experience

No installation required. Get started in 2 minutes

Install assemblies from NuGet

&

Download demo code from [GitHub](#) | Open demo code in [Visual Studio](#)

3. With a trial license, only the latest version's trial installer can be downloaded.
4. After downloading, the Syncfusion ASP . NET MVC - EJ2 trial installer can be unlocked using either the trial unlock key or the Syncfusion registered login credential. More information on generating an unlock key can be found in [this](#) article.
5. Before the trial expires, you can download the trial installer at any time from your registered account's [Trials & Downloads](#) page. (See the screenshot below)

Trial Downloads and Unlock Keys

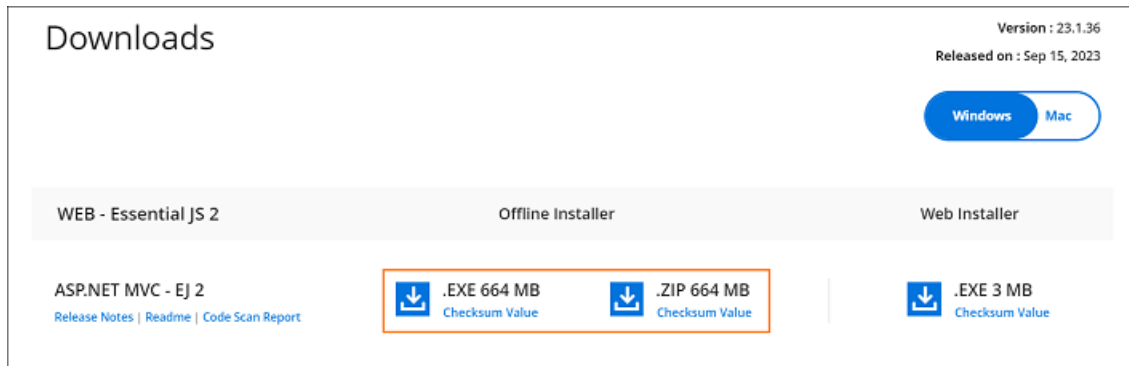
ASP.NET MVC (Essential JS 2)

Trial Version

[What's New](#) | [Release Notes](#) | [Get License Key](#) | [Get Unlock Key](#) | [Security Management Report](#)

Download	1
More Download Options	2

- Click the More Download Options (element 2 in above screenshot) button to get the Essential Studio ASP . NET MVC - EJ2 Offline trial installer which is available in EXE and ZIP format.



Start Trials if using components through [NuGet.org](https://www.nuget.org/packages?q=syncfusion)

You should initiate an evaluation if you have already obtained our components through [NuGet.org](https://www.nuget.org/packages?q=syncfusion)

- You can start your 30-day free trial for ASP . NET MVC - EJ2 from the [Start Trial](#) page from your account.



- To access this page, you must sign up/ log in with your Syncfusion account.
- Begin your trial by selecting the ASP . NET MVC - EJ2 product.

Note: If you've already used the trial products and they haven't expired, you won't be able to start the trial for the same product again.

- After you've started the trial, go to the [Trials & Downloads](#) page to get the latest version trial installer. You can generate the [unlock key](#) and [license key](#) here at any time before the trial period expires. (See the screenshot below)



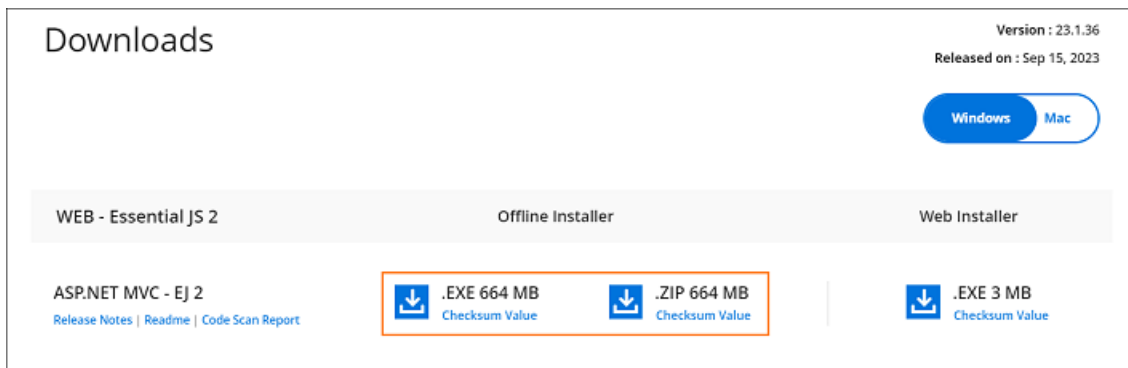
5. You can find your current active trial products on the [Trials & Downloads](#) page.

Download the License Version

1. Syncfusion licensed products will be available in the [License & Downloads](#) page under your registered Syncfusion account.
2. You can view all the licenses (both active and expired) associated with your account.
3. Click the Download (element 1 in the screenshot below) button to download the respective product's installer.
4. The most recent version of the installer will be downloaded from this page.
5. To download older version installers, go to [Downloads Older Versions](#) (element 2 in the screenshot below).
6. You can download other platform / add-on installers by going to More Downloads Options (element 3 in the screenshot below).



7. For Windows OS, EXE and Zip formats are available for download. They are both Offline Installers.



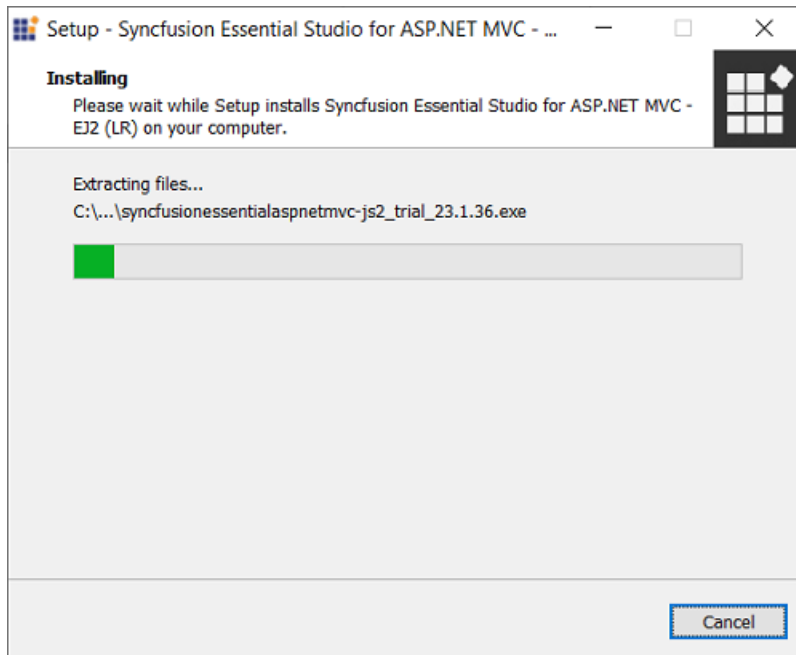
You can also refer to the [Offline Installer](#) links for step-by-step installation guidelines.

Installing Syncfusion ASP.NET MVC EJ2 Offline Installer

Installing with UI

The steps below show how to install the Essential Studio ASP . NET MVC - EJ2 installer.

1. Open the Syncfusion ASP . NET MVC - EJ2 offline installer file from downloaded location by double-clicking it. The Installer Wizard automatically opens and extracts the package.

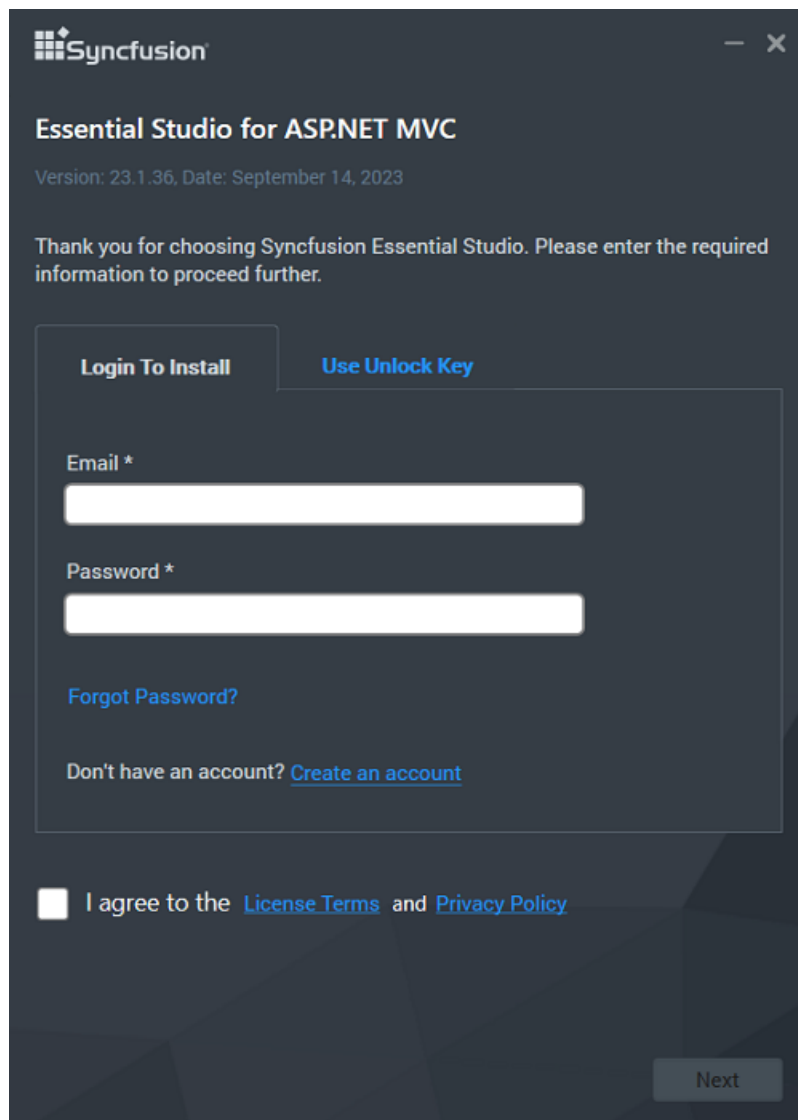


Note: The Installer wizard extracts the syncfusionessentialaspnetmvc-js2_(version).exe dialog, which displays the package's unzip operation.

2. To unlock the Syncfusion offline installer, you have two options:
 - *Login To Install*
 - *Use Unlock Key*

Login To Install

You must enter your Syncfusion email address and password. If you don't already have a Syncfusion account, you can sign up for one by clicking **“Create an account”**. If you have forgotten your password, click on **“Forgot Password”** to create a new one. Once you've entered your Syncfusion email and password, click Next.



Syncfusion

Essential Studio for ASP.NET MVC

Version: 23.1.36, Date: September 14, 2023

Thank you for choosing Syncfusion Essential Studio. Please enter the required information to proceed further.

Login To Install **Use Unlock Key**

Email *

Password *

[Forgot Password?](#)

Don't have an account? [Create an account](#)

☐ I agree to the [License Terms](#) and [Privacy Policy](#)

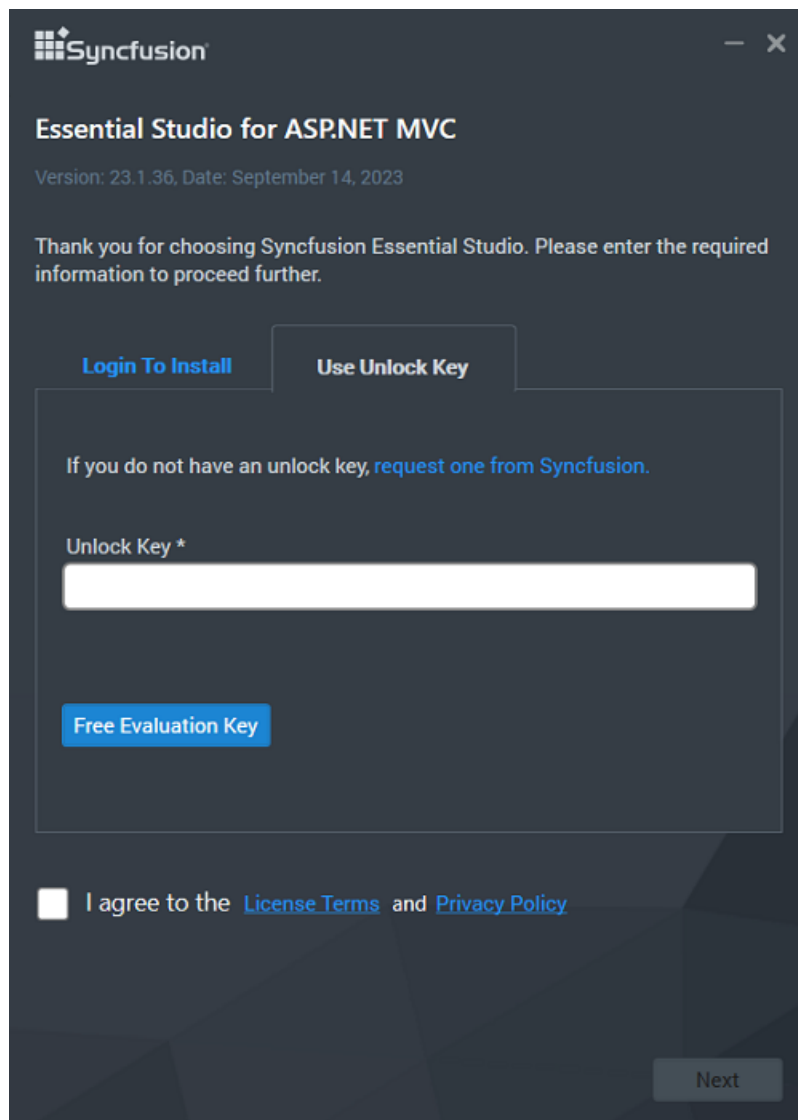
Next

Use Unlock Key

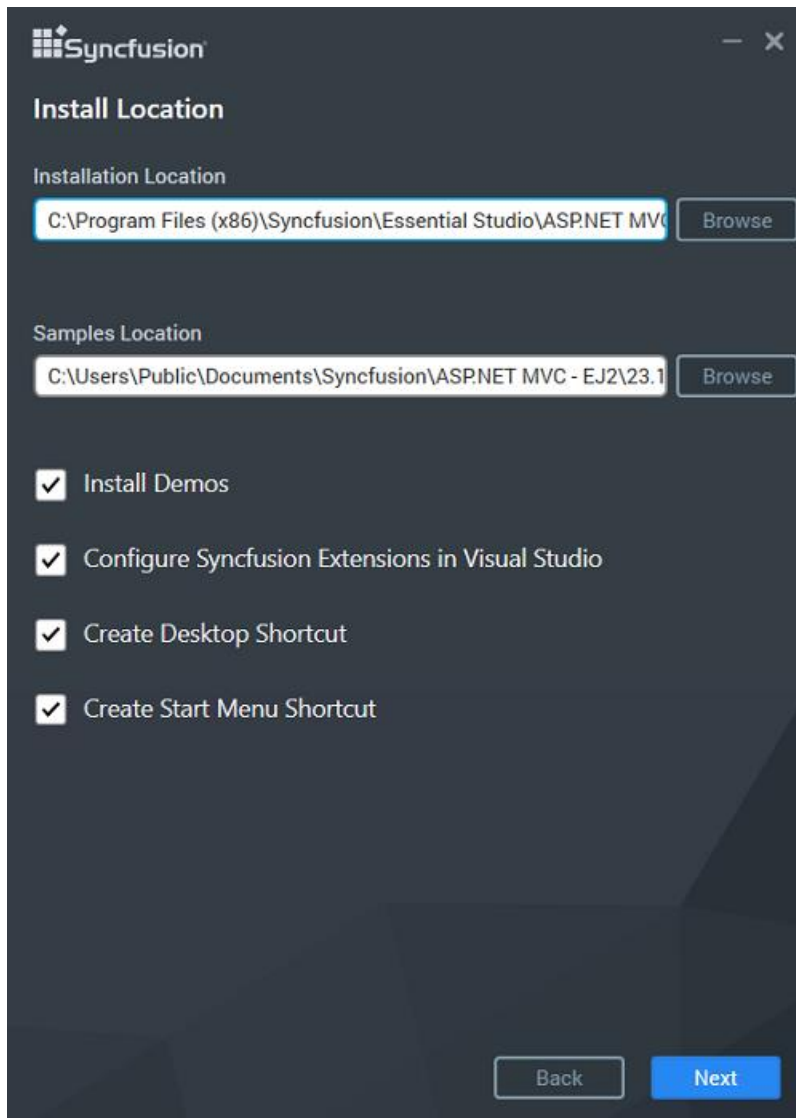
Unlock keys are used to unlock the Syncfusion offline installer, and they are platform and version specific. You should use either Syncfusion licensed or trial Unlock key to unlock Syncfusion ASP . NET MVC - EJ2 installer.

The trial unlock key is only valid for 30 days, and the installer will not accept an expired trial key.

To learn how to generate an unlock key for both trial and licensed products, see [this](#) Knowledge Base article.

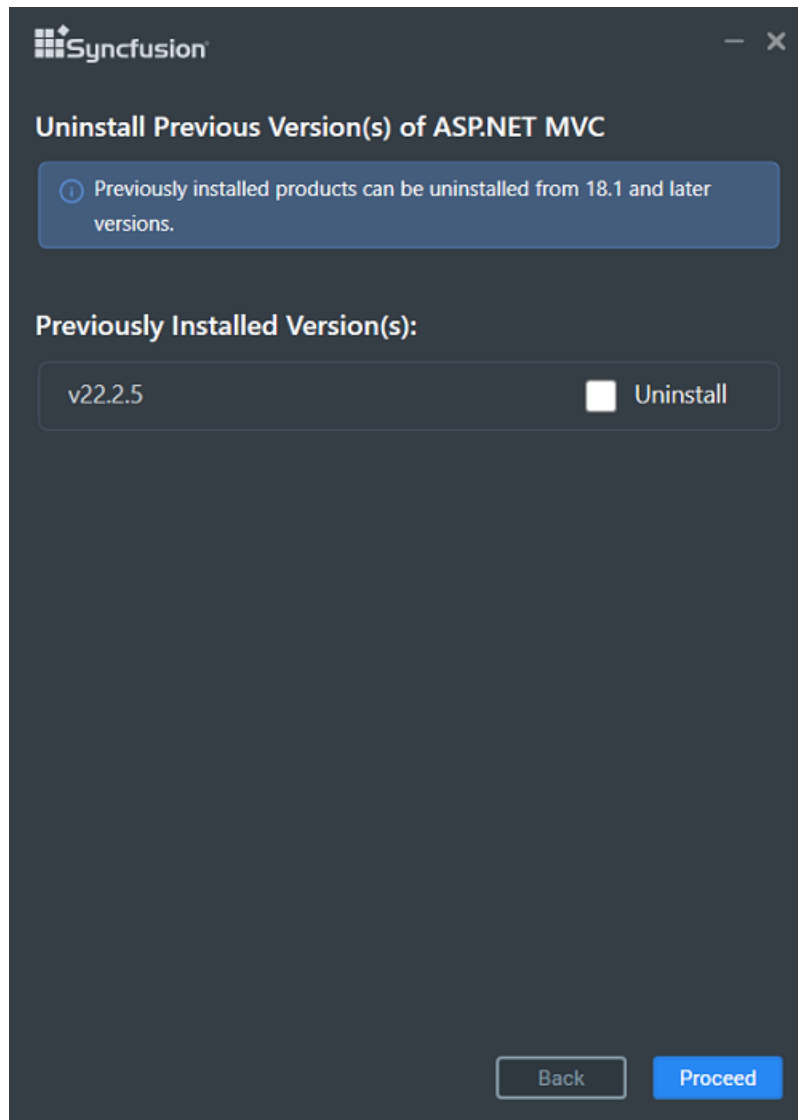


3. After reading the License Terms and Privacy Policy, check the **“I agree to the License Terms and Privacy Policy”** check box. Click the Next button.
4. Change the install and sample locations here. You can also change the Additional settings. Click Next/Install to install with the default settings.



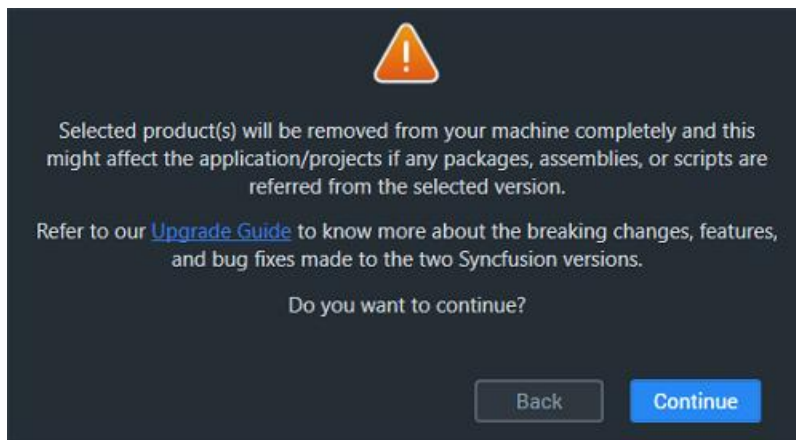
Additional Settings

- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box unchecked, if you don't want to install Syncfusion samples
 - Select the **Configure Syncfusion Extensions controls in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio
 - Check the **Create Desktop Shortcut** check box to add a desktop shortcut for Syncfusion Control Panel
 - Check the **Create Start Menu Shortcut** checkbox to add a shortcut to the start menu for Syncfusion Control Panel
5. If any previous versions of the current product is installed, the Uninstall Previous Version(s) wizard will be opened. Select **Uninstall** checkbox to uninstall the previous versions and then click the Proceed button.

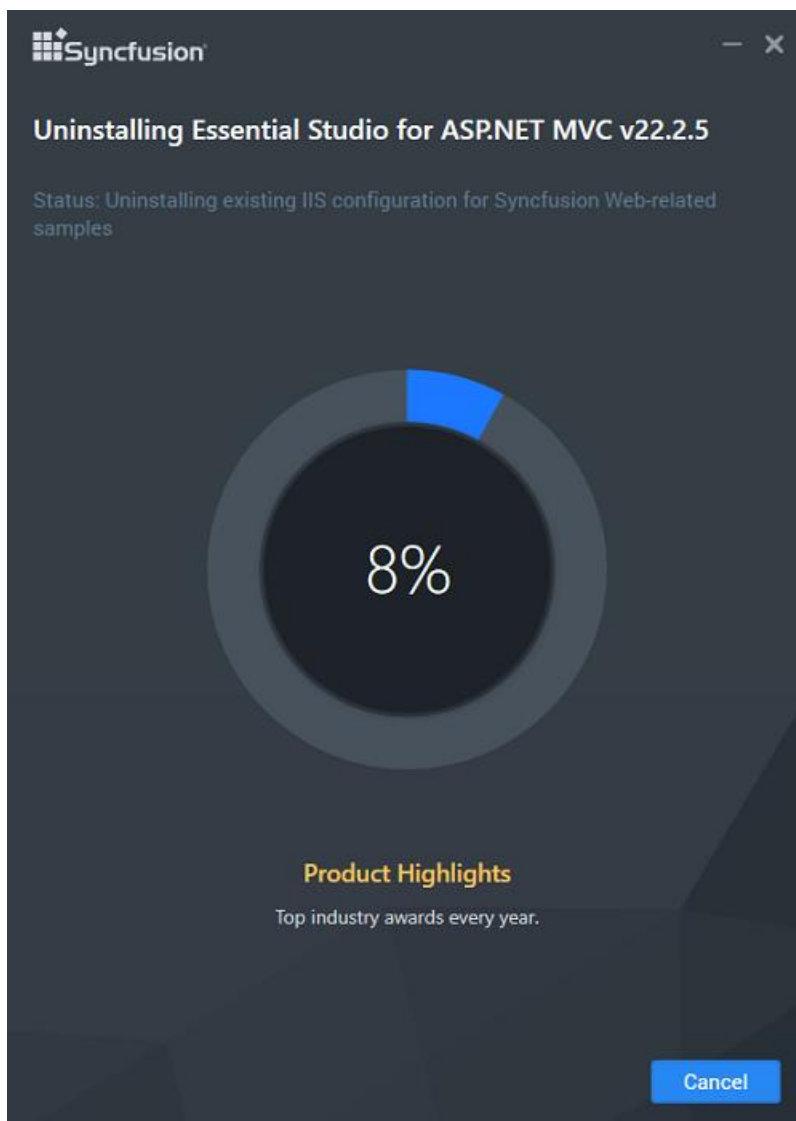


Note: From the 2021 Volume 1 release, Syncfusion has added the option to uninstall previous versions from 18.1 while installing the new version. If any version is selected to uninstall, a confirmation screen will appear; if continue is selected, the Progress screen will display the uninstall and install progress, respectively. If none of the versions are chosen to be uninstalled, only the installation progress will be displayed.

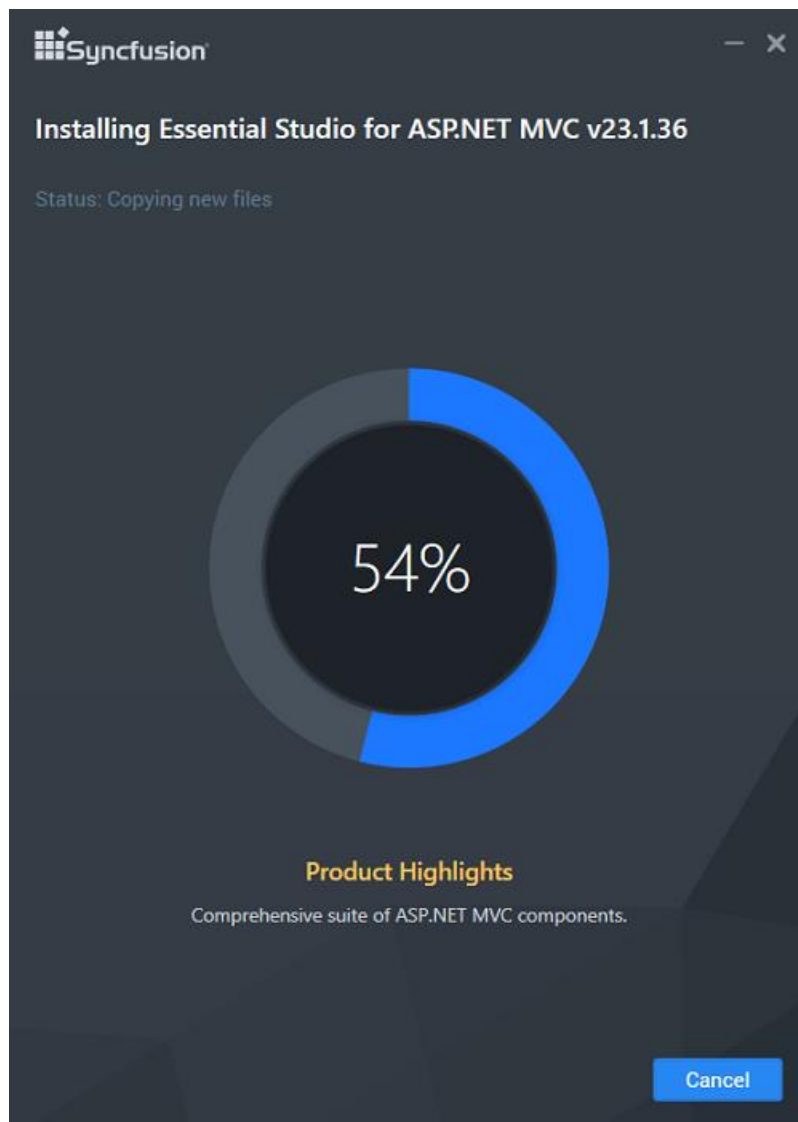
Confirmation Alert



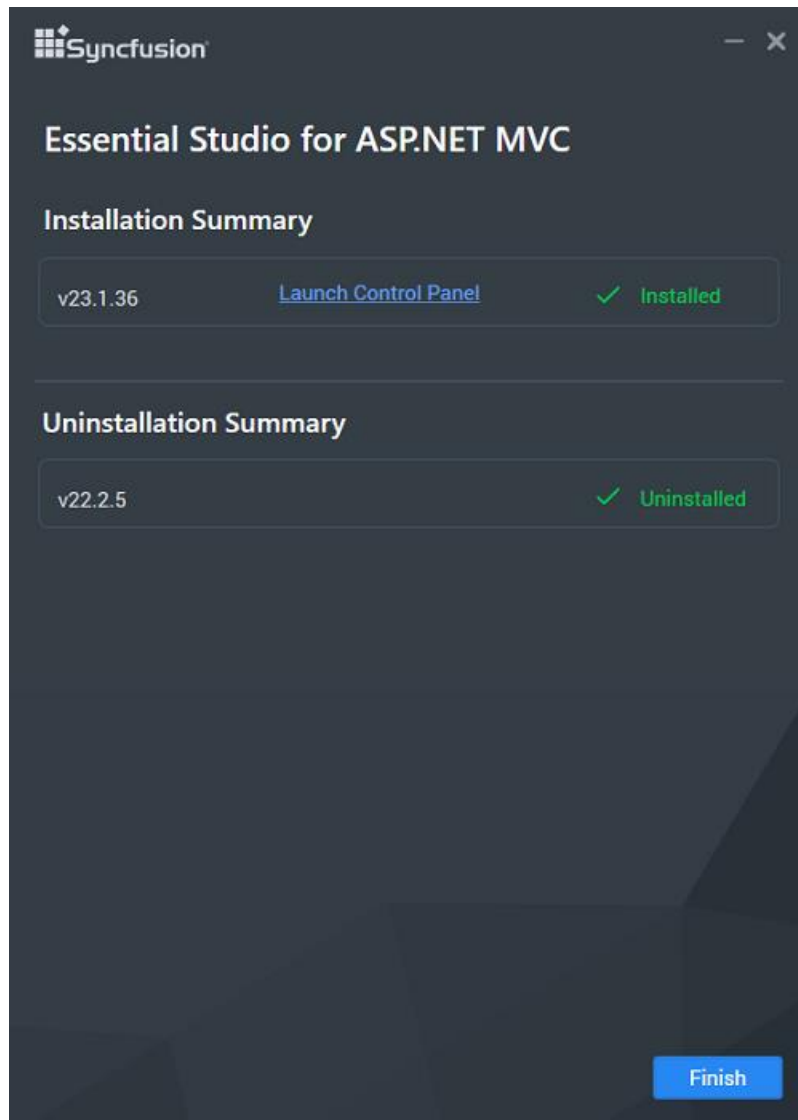
Uninstall Progress



Install Progress



Note: The Completed screen is displayed once the ASP . NET MVC - EJ2 product is installed. If any version is selected to uninstall, The completed screen will display both install and uninstall status.



6. After installing, click the **Launch Control Panel** link to open the Syncfusion Control Panel.
7. Click the Finish button. Your system has been installed with the Syncfusion Essential Studio ASP . NET MVC - EJ2 product.

Installing in silent mode

The Syncfusion Essential Studio ASP . NET MVC - EJ2 Installer supports installation and uninstallation via the command line.

Command Line Installation

To install through the Command Line in Silent mode, follow the steps below.

1. Run the Syncfusion ASP . NET MVC - EJ2 installer by double-clicking it. The Installer Wizard automatically opens and extracts the package.
2. The file syncfusionessentialaspnetmvc-js2_(version).exe file will be extracted into the Temp directory.

3. Run %temp%. The Temp folder will be opened. The syncfusionessentialaspnetmvc-js2_(version).exe file will be located in one of the folders.
4. Copy the extracted syncfusionessentialaspnetmvc-js2_(version).exe file in local drive.
5. Exit the Wizard.
6. Run Command Prompt in administrator mode and enter the following arguments.

Arguments: "installer file path\SyncfusionEssentialStudio(product)_(version).exe"/Install silent/UNLOCKKEY:"(product unlock key)"[/log"{Log file path}"][/InstallPath:{Location to install}]/[InstallSamples:{true/false}]/[InstallAssemblies:{true/false}]/[UninstallExistAssemblies:{true/false}]/[InstallToolbox:{true/false}]

Note: [...] - Arguments inside the square brackets are optional.

Example: "D:\Temp\syncfusionessentialaspnetmvc-js2x.x.x.x.exe"/Install silent/UNLOCKKEY:"*product unlock key*"/log "C:\Temp\EssentialStudioPlatform.log"/InstallPath:C:\Syncfusion\x.x.x.x /InstallSamples:true /InstallAssemblies:true /UninstallExistAssemblies:true /InstallToolbox:true

7. Essential Studio for ASP . NET MVC (Essential JS2) is installed.

Note: x.x.x.x should be replaced with the Essential Studio version and the Product Unlock Key needs to be replaced with the Unlock Key for that version.

Command Line Uninstallation

Syncfusion Essential ASP . NET MVC - EJ2 can be uninstalled silently using the Command Line.

1. Run the Syncfusion ASP . NET MVC - EJ2 installer by double-clicking it. The installer Wizard automatically opens and extracts the package.
2. The syncfusionessentialaspnetmvc-js2_(version).exe file will be extracted into the Temp directory.
3. Run %temp%. The Temp folder will be opened. The syncfusionessentialaspnetmvc-js2_(version).exe file will be located in one of the folders.
4. Copy the extracted syncfusionessentialaspnetmvc-js2_(version).exe file in local drive.
5. Exit the Wizard.
6. Run Command Prompt in administrator mode and enter the following arguments.

Arguments: "Copied installer file path\syncfusionessentialaspnetmvc-js2_(version).exe" /uninstall silent

Example: "D:\Temp\syncfusionessentialaspnetmvc-js2_x.x.x.x.exe" /uninstall silent

7. Essential Studio for ASP . NET MVC (Essentials JS2) is uninstalled.

Install Syncfusion ASP.NET MVC JS2 NuGet packages

Overview

NuGet is a Package management system for Visual Studio. It makes it easy to add, update and remove external libraries in our application. Syncfusion publishing all ASP.NET MVC JS2 NuGet packages in [nuget.org](https://www.nuget.org). The Syncfusion ASP.NET MVC JS2 NuGet packages can be used without installing the Syncfusion setup by Syncfusion Installer. You can simply exploit the Syncfusion ASP.NET MVC JS2 NuGet packages in ASP.NET MVC application to develop with the Syncfusion ASP.NET MVC JS2 components.

Note: The Syncfusion.EJ2.MVC5 NuGet package, which contains all Syncfusion ASP.NET MVC JS2 components in a single package, is available beginning with v16.1.0.24 (Essential Studio 2018 Volume 1).

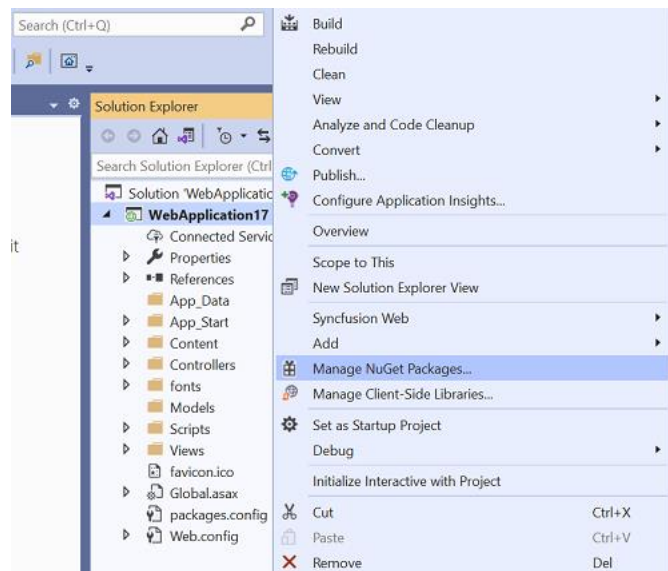
Installation using Package Manager UI

The NuGet **Package Manager UI** allows you to search, install, uninstall, and update Syncfusion ASP.NET MVC JS2 NuGet packages in your applications and solutions. You can find and install the Syncfusion ASP.NET MVC JS2 NuGet packages in your Visual Studio ASP.NET MVC web application and this process is easy with the steps below:

1. To open the Manage NuGet packages UI, follow either one of the options below:

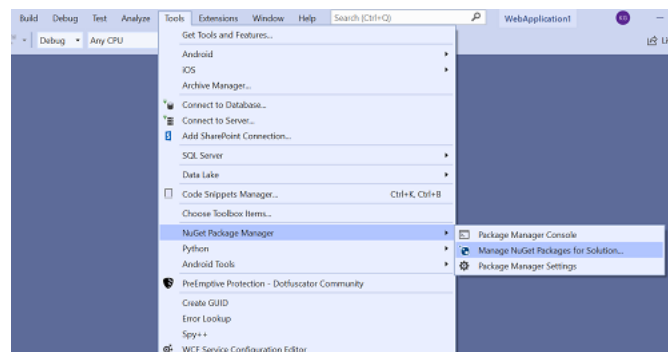
Option 1:

Right-click on the ASP.NET MVC web application or solution in the Solution Explorer, and choose **Manage NuGet Packages...**



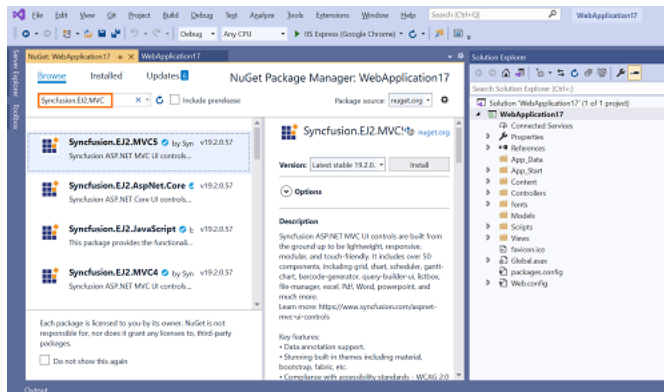
Option 2:

After opening the ASP.NET MVC web application in Visual Studio, go to the **Tools** menu and after hovering **NuGet Package Manager**, select **Manage NuGet Packages for Solution...**

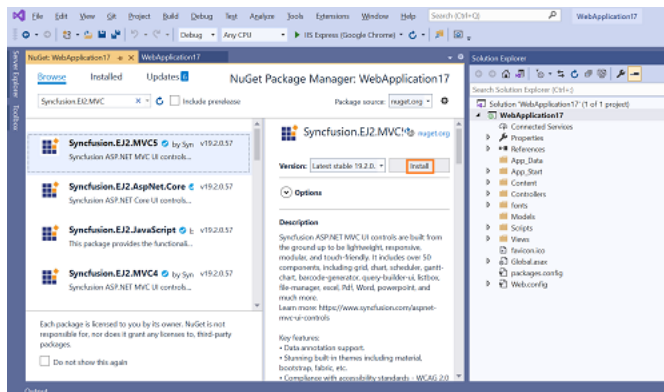


- The Manage NuGet Packages window will open. Navigate to the **Browse** tab, then search for the Syncfusion ASP.NET MVC JS2 NuGet packages using a term like “**Syncfusion.EJ2.MVC5**” and select the appropriate Syncfusion ASP.NET MVC NuGet package for your development.

Note: The nuget.org package source is selected by default in the Package source drop-down. If your Visual Studio does not have nuget.org configured, follow the instructions in the [Microsoft documents](#) to set up the nuget.org feed URL.



- When you select a package, the right side panel will provide more information about it.
- By default, the package selected with latest version. You can choose the required version and click the **Install** button and accept the license terms. The package will be added to your ASP.NET MVC application.

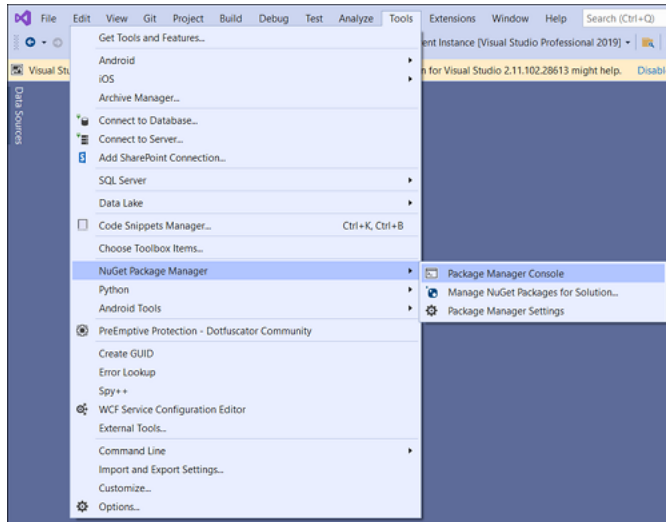


- At this point, your application has all the required Syncfusion assemblies, and you will be ready to start building high-performance, responsive app with [Syncfusion ASP.NET MVC JS2 components](#). Also, you can refer to the [ASP.NET MVC JS2 help document](#) for development.

Installation using Package Manager Console

The **Package Manager Console** saves NuGet packages installation time since you don't have to search for the Syncfusion.EJ2.MVC5 NuGet package which you want to install, and you can just type the installation command to install the appropriate Syncfusion ASP.NET MVC JS2 NuGet package. Follow the instructions below to use the Package Manager Console to reference the Syncfusion ASP.NET MVC JS2 component as NuGet packages in your ASP.NET MVC web application.

1. To show the Package Manager Console, open your ASP.NET MVC web application in Visual Studio and navigate to **Tools -> NuGet Package Manager -> Package Manager Console**.



2. The **Package Manager Console** will be shown at the bottom of the screen. You can install the Syncfusion ASP.NET MVC JS2 NuGet packages by enter the following NuGet installation commands.

Install specified Syncfusion ASP.NET MVC JS2 NuGet package.

The below command will install the Syncfusion ASP.NET MVC JS2 NuGet package in the default ASP.NET MVC application.

```
Install-Package <Package Name>
```

For example: Install-Package Syncfusion.EJ2.MVC5

Note: You can find the list of Syncfusion ASP.NET MVC JS2 NuGet packages which are published in nuget.org from [here](#)

Install specified Syncfusion ASP.NET MVC JS2 NuGet package in specified ASP.NET MVC application

The below command will install the Syncfusion ASP.NET MVC JS2 NuGet package in the given ASP.NET MVC application.

```
Install-Package <Package Name> -ProjectName <Project Name>
```

For example: Install-Package Syncfusion.EJ2.MVC5 -ProjectName SyncfusionMVCWebApplication

3. By default, the package will be installed with latest version. You can give the required version with the -Version term like below to install the Syncfusion ASP.NET MVC JS2 NuGet packages in the appropriate version.

```
Install-Package Syncfusion.EJ2.MVC5 -Version 19.2.0.57
```

```

PM> Install-Package Syncfusion.EJ2.MVC5 -Version 19.2.0.57

Attempting to gather dependency information for package 'Syncfusion.EJ2.MVC5.19.2.0.57' with respect to project 'SyncfusionMvcWebApplication', targeting '.NETFramework,Version=v4.7'
Gathering dependency information took 3.37 sec
Attempting to resolve dependencies for package 'Syncfusion.EJ2.MVC5.19.2.0.57' with DependencyBehavior 'lowest'
Resolving dependency information took 0 ms
Resolving actions to install package 'Syncfusion.EJ2.MVC5.19.2.0.57'
Resolved actions to install package 'Syncfusion.EJ2.MVC5.19.2.0.57'
Retrieving package 'Syncfusion.EJ2.JavaScript.19.2.0.57' from 'nuget.org'.
Retrieving package 'Syncfusion.EJ2.MVC5.19.2.0.57' from 'nuget.org'.
Retrieving package 'Syncfusion.Licensing.19.2.0.57' from 'nuget.org'.
GET https://api.nuget.org/v3-flatcontainer/syncfusion.ej2.mvc5/19.2.0.57/syncfusion.ej2.mvc5.19.2.0.57.nupkg
OK https://api.nuget.org/v3-flatcontainer/syncfusion.ej2.mvc5/19.2.0.57/syncfusion.ej2.mvc5.19.2.0.57.nupkg 26ms
Installing Syncfusion.EJ2.MVC5.19.2.0.57.
Adding package 'Syncfusion.EJ2.JavaScript.19.2.0.57' to folder 'C:\Users\KallirajanGanesan\source\repos\SyncfusionMvcWebApplication\packages'
Added package 'Syncfusion.EJ2.JavaScript.19.2.0.57' to folder 'C:\Users\KallirajanGanesan\source\repos\SyncfusionMvcWebApplication\packages'
Successfully installed 'Syncfusion.EJ2.JavaScript.19.2.0.57' to 'SyncfusionMvcWebApplication\packages'
Adding package 'Syncfusion.Licensing.19.2.0.57' to folder 'C:\Users\KallirajanGanesan\source\repos\SyncfusionMvcWebApplication\packages'
Added package 'Syncfusion.Licensing.19.2.0.57' to folder 'C:\Users\KallirajanGanesan\source\repos\SyncfusionMvcWebApplication\packages'
Successfully installed 'Syncfusion.Licensing.19.2.0.57' to 'SyncfusionMvcWebApplication\packages'
Adding package 'Syncfusion.EJ2.MVC5.19.2.0.57' to folder 'C:\Users\KallirajanGanesan\source\repos\SyncfusionMvcWebApplication\packages'
Added package 'Syncfusion.EJ2.MVC5.19.2.0.57' to folder 'C:\Users\KallirajanGanesan\source\repos\SyncfusionMvcWebApplication\packages'
Successfully installed 'Syncfusion.EJ2.MVC5.19.2.0.57' to 'SyncfusionMvcWebApplication\packages'
Executing nuget actions took 4.83 sec
File Elapsed: 00:00:09.5232315

```

4. The NuGet package manager console will install the Syncfusion ASP.NET MVC JS2 NuGet package as well as the dependencies it has. When the installation is complete, the console will show that your Syncfusion ASP.NET MVC JS2 package has been successfully added to the application.
5. At this point, your application has all the required Syncfusion assemblies, and you will be ready to start building high-performance, responsive app with [Syncfusion ASP.NET MVC JS2 components](#). Also, you can refer to the [ASP.NET MVC JS2 help document](#) for development.

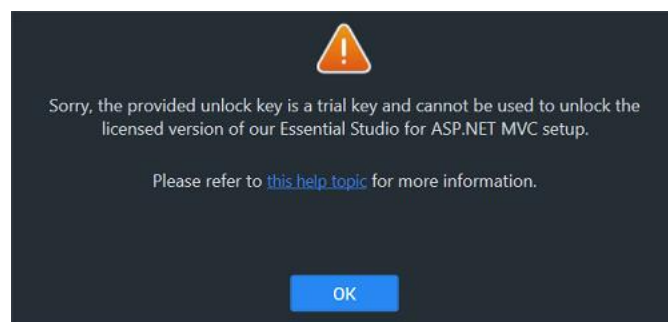
Common Installation Errors

This article describes the most common installation errors, as well as the causes and solutions to those errors.

- [Unlocking the license installer using the trial key](#)
- [License has expired](#)
- [Unable to find a valid license or trial](#)
- [Unable to install because of another installation](#)
- [Unable to install due to controlled folder access](#)

Unlocking the license installer using the trial key

Error Message: Sorry,the provided unlock key is a trial unlock key and cannot be used to unlock the licensed version of our Essential Studio for ASP . NET MVC installer.



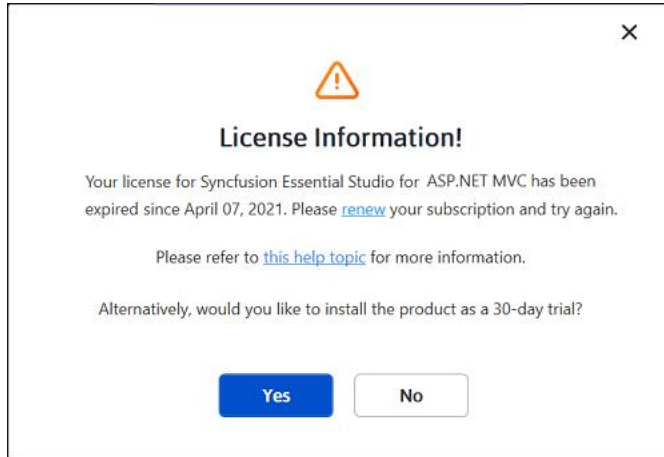
Reason
 You are attempting to use a Trial unlock key to unlock the licensed installer.

Suggested solution
 Only a licensed unlock key can unlock a licensed installer. So, to unlock the Licensed installer, use the Licensed unlock key. To generate the licensed unlock key, refer to [this](#) article.

License has expired

Error Message: Your license for Syncfusion Essential Studio for ASP . NET MVC - EJ2 has been expired since {date}. Renew your subscription and try again.

Online Installer



Reason
 This error message will appear if your license has expired.

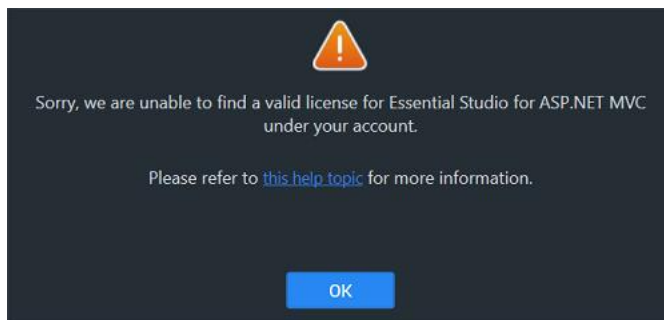
Suggested Solution
 You can choose from the options below.

1. You can renew your subscription [here](#).
2. You can get a new license [here](#).
3. You can reach out to our sales team by emailing sales@syncfusion.com.
4. You can also extend the 30-day trial period after your trial license has expired.

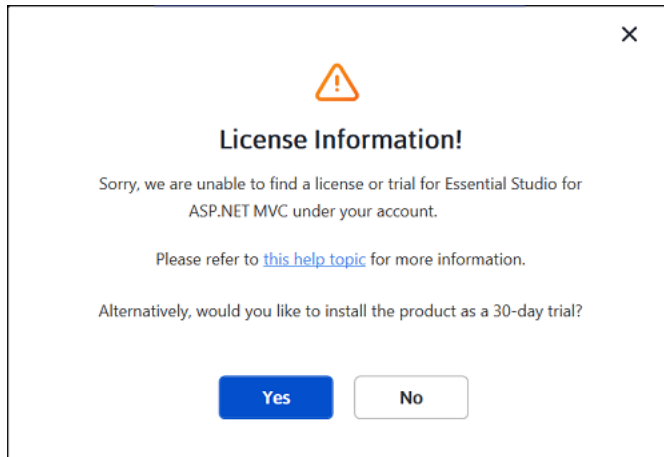
Unable to find a valid license or trial

Error Message: Sorry, we are unable to find a valid license or trial for Essential Studio for ASP . NET MVC - EJ2 under your account.

Offline installer



Online installer



Reason
 The following are possible causes of this error:

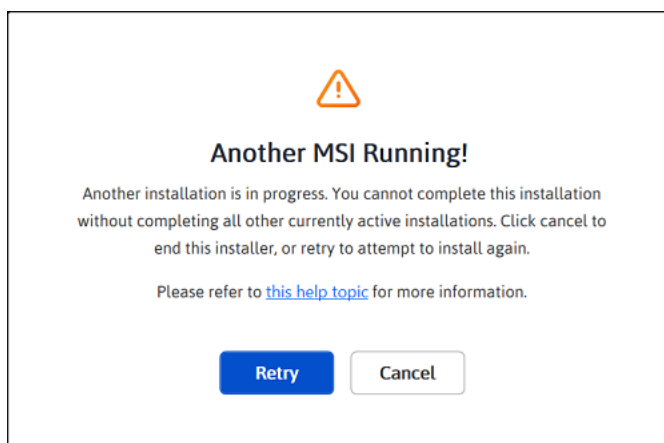
- When your trial period expired.
- When you don't have a license or an active trial.
- You are not the license holder of your license.
- Your account administrator has not yet assigned you a license.

Suggested solution

1. You can get a new license [here](#).
2. Contact your account administrator.
3. Send an email to clientrelations@syncfusion.com to request a license.
4. You can reach out to our sales team by emailing sales@syncfusion.com.

Unable to install because of another installation

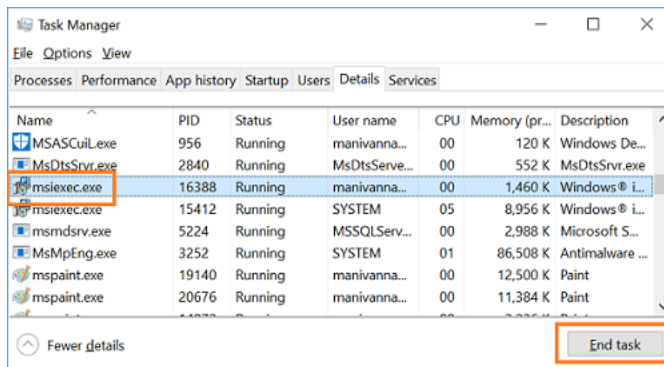
Error Message: Another installation is in progress. You cannot start this installation without completing all other currently active installations. Click cancel to end this installer or retry to attempt after currently active installation completed to install again.



Reason
 You are trying to install when another installation is already running in your machine.

Suggested solution
 Open and kill the msixec process in the task manager and then continue to install Syncfusion. If the problem is still present, restart the computer and try Syncfusion installer.

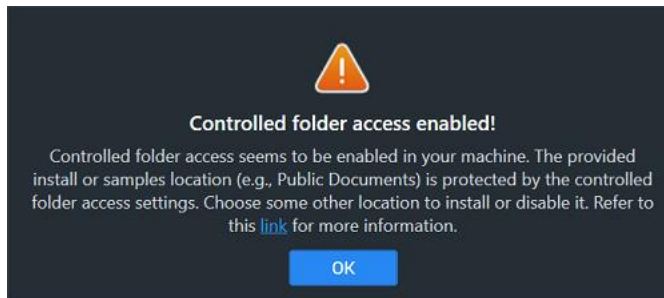
1. Open the Windows Task Manager.
2. Browse the Details tab.
3. Select the `msiexec.exe` and click **End Task**.



Unable to install due to controlled folder access

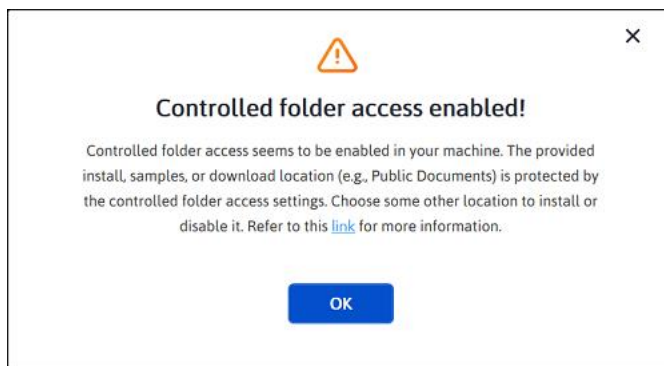
Offline:

Error Message: Controlled folder access seems to be enabled in your machine. The provided install or samples location (e.g., Public Documents) is protected by the controlled folder access settings.



Online:

Error Message: Controlled folder access seems to be enabled in your machine. The provided install, samples, or download location (e.g., Public Documents) is protected by the controlled folder access settings.



Reason
 You have enabled controlled folder access settings on your computer.

Suggested solution

Suggestion 1:
 1. We will ship our demos in the public documents folder by default.

2. You have controlled folder access enabled on your machine, so our demos cannot be installed in the documents folder. If you need to install our demos in the Documents folder, follow the steps in this [link](#) and disable the controlled folder access. 3. You can enable this option after the installing our Syncfusion setup.

Suggestion 2: 1. If you do not want to disable controlled folder access, you can install our demos in another directory.

Licensing

Syncfusion Licensing Overview

We have introduced a new licensing system starting with version 16.2.0.x release of Essential Studio. These modifications apply to all evaluators and only to paid customers who use NuGet packages from [nuget.org](#). Starting with v16.2.0.x, if you use the evaluation installer or the NuGet feed to reference Syncfusion assemblies, you must also include the corresponding platform and version license key in your projects.

Difference between unlock key and license key

Note that this license key is different from the installer unlock key that you might have used in the past and needs to be separately generated from Syncfusion website. Refer [this](#) KB article to know more about difference between the Syncfusion Unlock Key and the Syncfusion License Key.

Following licensing error will be shown if the license key is not registered in your projects, while using assemblies from evaluation installer or from the [nuget.org](#)

Note: This application was built using a trial version of Syncfusion Essential Studio. Include a valid license to permanently remove this license validation message. You can also obtain a free 30 day evaluation license to temporarily remove this message during the evaluation period. Refer to this [help topic](#) for more information.

Registering Syncfusion license keys in Build server

| Source of Syncfusion assemblies | Details | License Key needs to be registered? | Where to get license key from |

| ----- | ----- | ----- | ----- |

| **NuGet package** | If the Syncfusion assemblies used in Build Server were from the Syncfusion NuGet packages, then no need to install any Syncfusion installer. We can directly use the required Syncfusion NuGet packages at [nuget.org](#).

But, if using NuGet packages from the [nuget.org](#), then we should register the Syncfusion license key in the application. | Yes | Use any developer license to [generate](#) keys for Build Environments as well. |

| **Trial installer** | If the Syncfusion assemblies used in Build Server were from Trial Installer, we should register the license key in the application for the corresponding version and platforms, to avoid trial license warning. | Yes | Use any developer trial license to [generate](#) keys for Build Environments as well. |

| **Licensed installer** | If the Syncfusion assemblies used in Build Server were from Licensed Installer, then there is no need to register the license keys.

You can [download](#) and [install](#) the licensed version of our installer. | No | Not applicable |

See Also

- [How to Generate Syncfusion ASP .NET MVC EJ2 License Key?](#)
- [How to Register Syncfusion License Key in ASP .NET MVC EJ2 Application?](#)

Generate Syncfusion ASP .NET MVC EJ2 License key

License keys can be generated from the [License & Downloads](#) or [Trial & Downloads](#) section of the Syncfusion website.

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio[Download Older Versions](#)

Latest Official Release : 23.1.36 (Volume 3 2023 - September 15, 2023)

Download

[What's New](#) | [Release Notes](#) | [Get License Key](#) ⓘ | [Get Unlock Key](#) ⓘ | [Code Scan Report](#)[More Download Options](#)

Syncfusion license keys are **version and platform specific**, refer to the [KB](#) to generate the license key for the required version and platform. Also, refer this [KB](#) to know about which version of the Syncfusion license key should be used in the application.

Claim License key

Syncfusion License keys can also be generated from the “**Claim License Key**” page based on the trial or valid license associated with your Syncfusion account.

You can get the license key, based on license availability in your Syncfusion account.

Active License

If you have a Syncfusion account associated with valid license, license key will be generated from claim license key page.

Claim License Key

Version : 23.1.36 ▾

Essential Studio Enterprise Edition v23.1.36

COPY TO CLIPBOARD

SEND EMAIL

For more details about this generated key, [click here](#).

Note: A license key cannot be generated for our Java platform. If you are looking for a Java license key, please [click here](#).

Please refer to this [help topic](#) to learn how to register your license key.

Active Trial

If you have a Syncfusion account associated with valid trial license, license key will be generated from claim license key page with expiry date.

Claim License Key

Version : 23.1.36

Essential Studio Enterprise Edition

Trial v23.1.36

[COPY TO CLIPBOARD](#) [SEND EMAIL](#)

Note: A license key cannot be generated for our java platform. If you are looking for a java license key, please [click here](#).

Your license key expires on November 23, 2023.

Please refer to this [help topic](#) to learn how to register your license key.

If you are looking for valid licenses, try one of the below options.

Purchase Essential Studio

The world's best UI component suite for building powerful web, desktop, and mobile apps.

[BUY NOW](#)

Free Community License

Eligibility: Companies and individuals with less than \$1 million USD in annual gross revenue and 5 or fewer developers.

[CLAIM YOUR FREE LICENSE](#)

Expired License

If you have a Syncfusion account with an expired license, your license subscription must be renewed in order to obtain a valid license key for the latest Essential Studio version. Meanwhile, a temporary license key with a 5-day validity period will be generated.

Your Essential Studio for ASP.NET license has expired.

Please renew your subscription to get a valid license key. You can renew online or reach our renewal team.

[CONTACT SYNCFUSION](#) [RENEW ONLINE](#)

For your convenience, Syncfusion has provided a temporary license key with 5 days of validity.

Essential Studio Enterprise Edition

Trial v23.1.36

[COPY TO CLIPBOARD](#) [SEND EMAIL](#)

Note: A license key cannot be generated for our java platform. If you are looking for a java license key, please [click here](#).

Your license key expires on October 29, 2023.

Please refer to this [help topic](#) to learn how to register your license key.

No Trial or No License or Expired trial

If the Syncfusion account is not associated with a trial, license, or expired trial, you can try to claim either a trial or a valid license from claim license page.

No active trial or license is currently associated with this account.
(Click here to learn why.)

For your convenience, Syncfusion has generated a 30-day free trial license key. Check the license key details below.

Essential Studio Enterprise Edition Trial v23.1.36

[COPY TO CLIPBOARD](#) [SEND EMAIL](#)

Note: A license key cannot be generated for our Java platform. If you are looking for a Java license key, please [click here](#).

Your license key expires on November 23, 2023.

Please refer to this [help topic](#) to learn how to register your license key.

If you are looking for valid licenses, try one of the below options.

Purchase Essential Studio
The world's best UI component suite for building powerful web, desktop, and mobile apps.
[BUY NOW](#)

Free Community License
Eligibility: Companies and individuals with less than \$1 million USD in annual gross revenue and 5 or fewer developers.
[CLAIM YOUR FREE LICENSE](#)

See Also

- [How to Register Syncfusion License Key in ASP .NET MVC EJ2 Application?](#)

Register Syncfusion License key in ASP .NET MVC EJ2 application

License key should be registered, if your project using Syncfusion ASP .NET MVC - EJ2 packages reference from nuget.org or from syncfusion installer.

ASP .NET MVC

Register the license key in Application_Start method of **Global.asax.cs/Global.asax.vb**

```
C# VB.NET

void Application_Start(object sender, EventArgs e)
{
    //Register Syncfusion License
    Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense("YOUR LICENSE KEY");

    // Code that runs on application startup
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

Syncfusion Licensing Errors

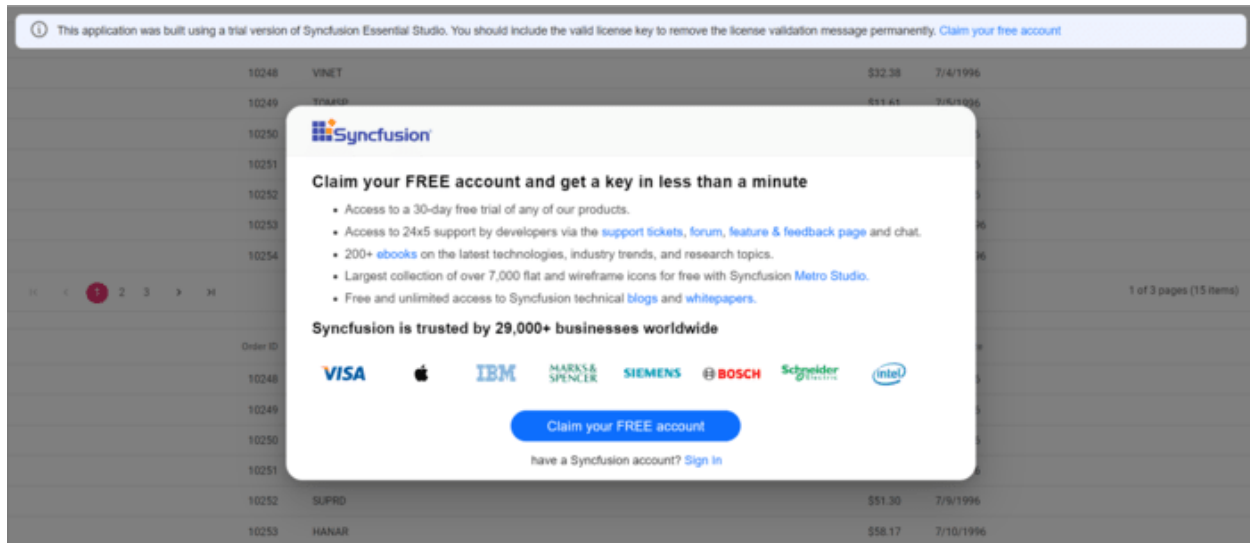
Licensing error pop up is displayed with various messages under different circumstances. Here are some ways to resolve different issues.

Licensing errors

License key not registered/Trial Expired

The following error message will be shown if a Syncfusion license key has not been registered in your application or if the trial key has expired after 30 days.

Error message : This application was built using a trial version of SynCFusion Essential Studio. You should include the valid license key to remove the license validation message permanently.



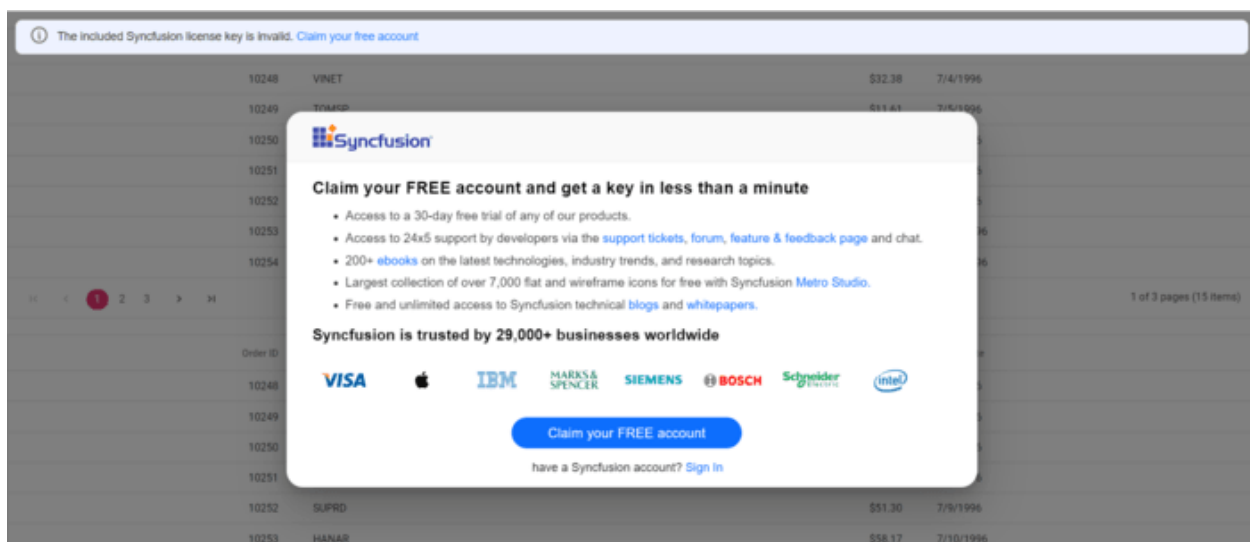
Solution:

- Generate a valid license key from here [Licensed users](#) or [Trial users](#) for a specific version and platform.
- Also, you can generate the license key from claim license key page by clicking the “**Claim your FREE account**” click from the licensing warning message. Refer to this [help topic](#) for more details.
- In your application, register the generated license key. Refer to this [help topic](#) for information on registering the license key.

Invalid key

If the application is registered with an invalid key, another version of license key, or another platform's license key, the following error message will pop up when launching the application.

Error Message: The included SynCFusion license key is invalid.



Solution:

- Generate a valid license key from here [Licensed users](#) or [Trial users](#) for a specific version and platform.
- Also, you can generate the license key from claim license key page by clicking the “**Claim your FREE account**” click from the licensing warning message. Refer to this [help topic](#) for more details.
- In your application, register the generated license key. Refer to this [help topic](#) for information on registering the license key.

Licensing errors from version 16.2.0* to 20.3.0**License key not registered*

The following error message will be shown if a Syncfusion license key has not been registered in your application.

Error message : This application was built using a trial version of Syncfusion Essential Studio. Include a valid license to permanently remove this license validation message. You can also obtain a free 30 day evaluation license to temporarily remove this message during the evaluation period. Refer to this [help topic](#) for more information.

Solution:

- Generate a valid license key from here [Licensed users](#) or [Trial users](#) for a specific version and platform.
- In your application, register the generated license key. Refer to this [help topic](#) for information on registering the license key.

Invalid key

If the application is registered with an invalid key, another version of license key, or another platform's license key, the following error message will pop up when launching the application.

Error Message: The included Syncfusion license is invalid. Refer to this [help topic](#) for more information.

Solution:

- Generate a valid license key from here [Licensed users](#) or [Trial users](#) for a specific version and platform.
- In your application, register the generated license key. Refer to this [help topic](#) for information on registering the license key.

Trial Expired

The following error message will be shown if the trial key has expired after 30 days

Error Message: Your Syncfusion trial license has expired. Refer to this [help topic](#) for more information.

Solution: Purchase from [here](#) to get a valid Syncfusion license.

Platform Mismatch

If the application is registered with another platform's license key, the following error message will pop up when launching the application.

Error Message: The included Syncfusion license is invalid (Platform mismatch). Refer to this [help topic](#) for more information.

Solution:

- Generate a valid license key from here [Licensed users](#) or [Trial users](#) for a specific version and platform.
- In your application, register the generated license key. Refer to this [help topic](#) for information on registering the license key.

Version Mismatch

If the application is registered with another version's license key, the following error message will pop up when launching the application.

Error Message: The included Syncfusion license ({Registered Version}) is invalid for version {Required Version}. Refer to this [help topic](#) for more information.

Solution:

- Generate a valid license key from here [Licensed users](#) or [Trial users](#) for a specific version and platform. Follow the [KB](#) to generate license key.
- In your application, register the generated license key. Refer to this [help topic](#) for information on registering the license key.

Could not load Syncfusion.Licensing.dll assembly version

Make sure that all the referenced Syncfusion assemblies are of same version. Try cleaning and rebuilding the application to resolve assembly conflict issues.

Syncfusion Licensing FAQ

Is an internet connection required for Syncfusion Essential Studio license validation?

- Syncfusion license validation is done offline during application execution and does not require internet access.
- Apps registered with a Syncfusion license key can be deployed on any system that does not have an internet connection.

How to upgrade from Trial version after purchasing a license

To upgrade from trial version, there are two possible solutions.

- Uninstall the trial version and install the fully licensed build from the [License & Downloads](#) section of our website.
- If you are using Syncfusion controls from [nuget.org](#), replace the currently used trial license key with a paid license key that can be generated from the [License & Downloads](#) section of our website. Refer to [this](#) topic for more information regarding registering the license in the application.

Note: License registration is not required if you reference Syncfusion assemblies from Licensed installer. These licensing changes applicable to all evaluators who refers the Syncfusion assemblies from evaluation installer and those who use Syncfusion NuGet packages from [nuget.org](#)

Where can I get a License key

License keys can be generated from the [License & Downloads](#) or [Trial & Downloads](#) section of the Syncfusion website.

Essential Studio Enterprise Edition - Community license	
Essential Studio Enterprise Edition Binary with TestStudio	Download Older Versions
Latest Official Release : 23.1.36 (Volume 3 2023 - September 15, 2023)	Download
What's New Release Notes Get License Key ⓘ Get Unlock Key ⓘ Code Scan Report	More Download Options

Syncfusion license keys are **version and platform specific**, refer to the [KB](#) to generate the license key for the required version and platform. Also, refer this [KB](#) to know about which version of the Syncfusion license key should be used in the application.

Registering Syncfusion account for direct NuGet.org user

If you have directly obtained Syncfusion assemblies from [NuGet.org](#) and do not have a Syncfusion account, follow the steps to obtain a free 30-day trial license key:

- Register for a free Syncfusion account [here](#).
- Go to the start trials [page](#) and start a trial.
- Finally proceed to the [Trials & Downloads](#) section to obtain the [license key](#).

Registering license key for Syncfusion JavaScript Components used in ASP.NET MVC App

If you are using [Syncfusion JavaScript Components](#) in your application, then you have to register license for ASP.NET MVC in `Global.asax.cs` and for JavaScript components register license in `_Layout.cshtml` after referring to the Syncfusion styles and scripts, as follows:

Note: From 2022 Vol 1 (v20.1) only the license key registration required for Syncfusion Javascript Components.

GLOBAL.ASAX.CS

```
protected void Application_Start()
{
    //Register Syncfusion license
    Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense("YOUR LICENSE KEY");
    AreaRegistration.RegisterAllAreas();
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
    site.ej2version }}/fluent.css />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src= https://cdn.syncfusion.com/ej2/{{ site.ej2version
  }}/dist/ej2.min.js></script>
<script>
  // Registering Syncfusion license key
  ej.base.registerLicense('License Key');
</script>
</head>
```

Upgrade

Upgrading Syncfusion ASP .NET MVC EJ2 installer to a latest version

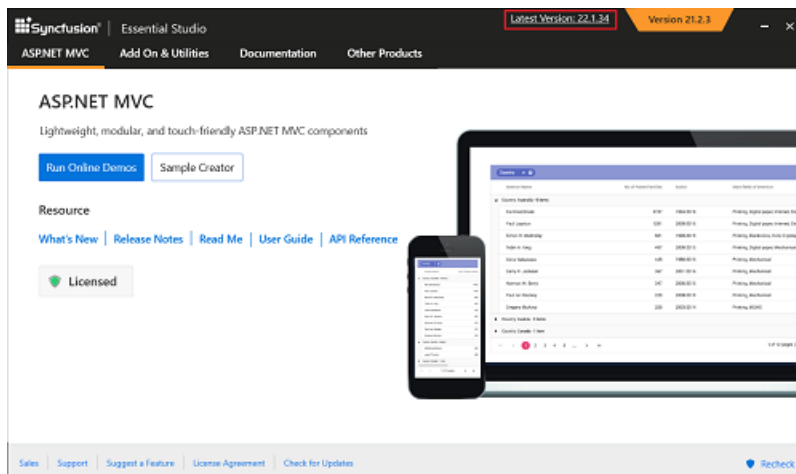
Syncfusion releases new volumes once every three months, with exciting new features. There will be one Service Pack release for these volume releases. Service Pack releases are provided to address major bug fixes in the volume releases.

You can upgrade to our latest version from any installed Syncfusion version.

See our "[Upgrade Guide](#)" for ASP . NET MVC - EJ2 to learn more about the "Breaking Changes, Bug Fixes, Features and Known Issues" between your current version and the latest version you are trying to upgrade.

Upgrading to the latest version

The most recent version of Syncfusion ASP . NET MVC - EJ2 can be downloaded and installed by clicking on the "Latest Version: {Version}" link at the top of the Syncfusion ASP . NET MVC - EJ2 Control Panel.



You can also upgrade to the latest version just by downloading and installing the products you require from [this](#) link. The existing installed versions are not required to be uninstalled.

It is not required to install the Volume release before installing the Service Pack release. As releases for Volume and Service Packs work independently, you can install the latest version with major bug fixes directly.

Upgrading from trial version to license version

To upgrade from trial version, there are two possible solutions.

- Uninstall the trial version and install the fully licensed build from the [License & Downloads](#) section of our website.

- If you are using Syncfusion controls from nuget.org, replace the currently used trial license key with a paid license key that can be generated from the [License & Downloads](#) section of our website. Refer to [this](#) topic for more information regarding registering the license in the application.

Note: License registration is not required if you reference Syncfusion assemblies from Licensed installer. These licensing changes applicable to all evaluators who refers the Syncfusion assemblies from evaluation installer and those who use Syncfusion NuGet packages from nuget.org.

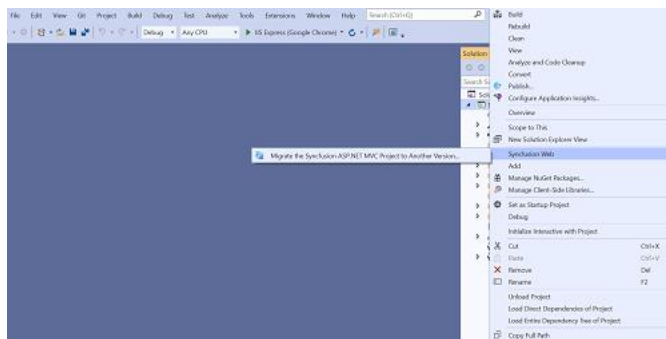
Upgrading Syncfusion ASP.NET MVC JS2 components to latest version

The Syncfusion ASP.NET MVC JS2 migration add-in for Visual Studio allows you to migrate an existing Syncfusion ASP.NET MVC web application from one version of Essential Studio version to another version. This reduces the amount of manual work required when migrating the Syncfusion version.

The steps below will assist you to upgrade the Syncfusion version in the Syncfusion ASP.NET MVC application via Visual Studio 2019:

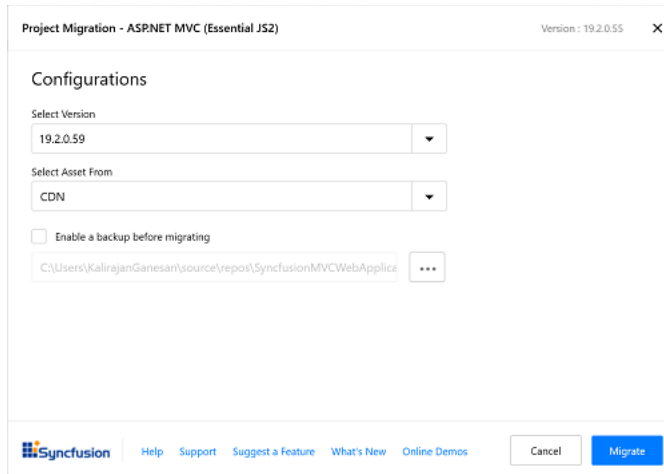
Note: Before use the Syncfusion ASP.NET MVC Web application Migration, check whether the Syncfusion ASP.NET MVC JS2 Extension installed or not in Visual Studio Extension Manager by clicking on the Extensions -> Manage Extensions -> Installed. If this extension not installed, install the extension by follow the steps from the [download and installation](#) help topic.

1. Open the Syncfusion ASP.NET MVC Web application that uses the Syncfusion component.
2. Open the Migration Wizard, by right-click the **Syncfusion ASP.NET MVC Application** from Solution Explorer and select **Syncfusion Web**. Choose the **Migrate Syncfusion ASP.NET MVC project from another version...**



3. The Syncfusion Project Migration window will appear. You can choose the required version of Syncfusion ASP.NET MVC to migrate.

Note: The versions are loaded from the Syncfusion ASP.NET MVC NuGet packages published in [NuGet.org](https://nuget.org) and it requires internet connectivity.



4. Click **Migrate** button, then the Syncfusion.EJ2.MVC5 NuGet packages will be updated to the respective chosen version in the application.
5. If you installed the trial setup or NuGet packages from nuget.org you must register the Syncfusion license key to your application since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Navigate to the [help topic](#) to generate and register the Syncfusion license key to your application. Refer to this [blog](#) post for understanding the licensing changes introduced in Essential Studio.

Upgrading Syncfusion ASP.NET MVC NuGet packages to a latest version

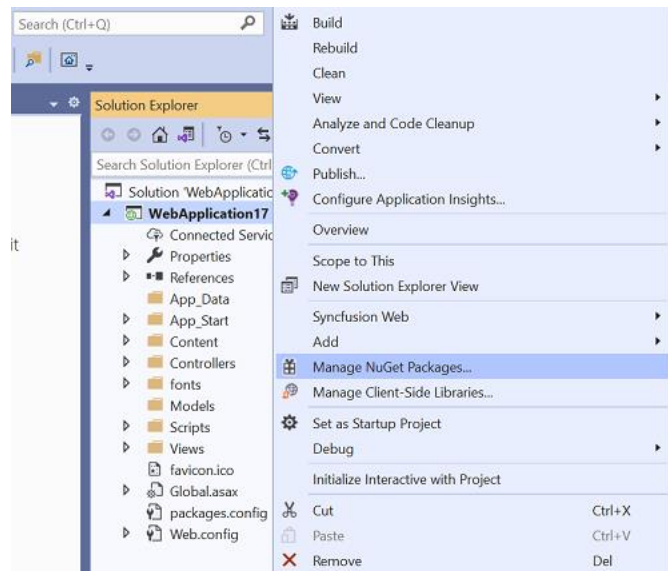
Every three months, Syncfusion releases new volumes with interesting new features. For this volume, there will be weekly NuGet releases and a service pack. Syncfusion ASP.NET MVC JS2 NuGet packages are released on a weekly basis to address critical issue fixes.

From any Syncfusion ASP.NET MVC JS2 NuGet version you have installed; you can update to our most recent version.

Upgrade NuGet packages through Package Manager UI

The NuGet **Package Manager UI** in Visual Studio allows you to easily install, uninstall, and update NuGet packages in applications and solutions. You can find and upgrade the Syncfusion ASP.NET MVC JS2 NuGet packages to the most recent version or to specific version in the ASP.NET MVC solution or application and this process is easy with the steps below:

1. Right-click on the ASP.NET MVC application or solution in the Solution Explorer tab, and choose **Manage NuGet Packages...**

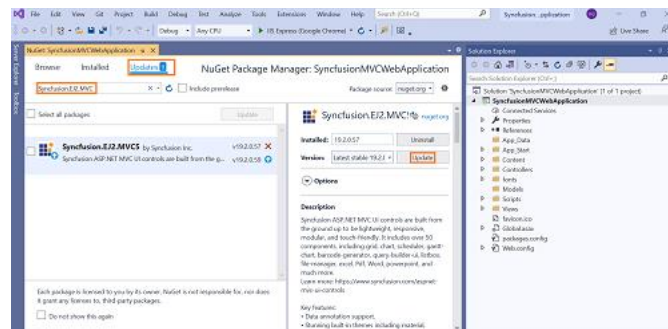


As an alternative, after opening the ASP.NET MVC application in Visual Studio, go to the **Tools** menu and after hovering **NuGet Package Manager**, select **Manage NuGet Packages for Solution...**

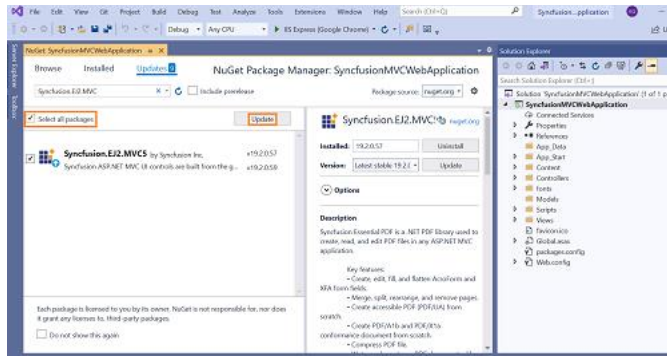
2. The Manage NuGet Packages window will open. Navigate to the **Updates** tab, then search for the Syncfusion ASP.NET MVC JS2 NuGet packages using a term like **"Syncfusion"** and select the appropriate Syncfusion ASP.NET MVC JS2 NuGet package for your application.

Note: The nuget.org package source is selected by default in the Package source drop-down. If your Visual Studio does not have nuget.org configured, follow the instructions in the [Microsoft documents](#) to set up the nuget.org feed URL.

3. By default, the package selected with latest version. You can select the required version and click the **Update** button and accept the license terms. The package will be upgraded to selected version in your ASP.NET MVC application.



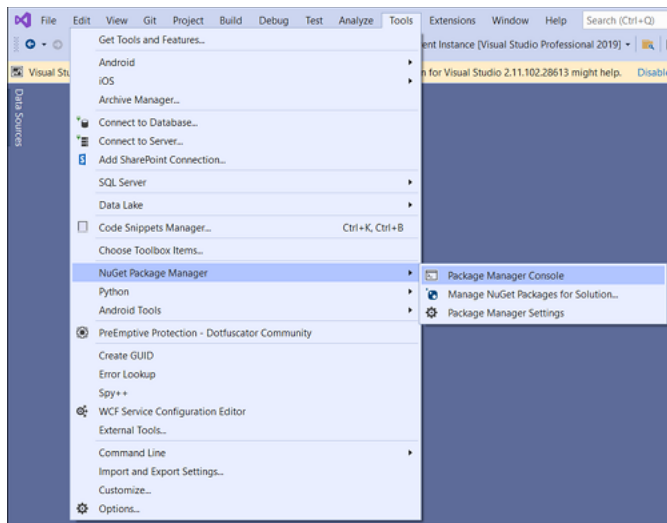
You can choose the multiple NuGet packages by selecting the checkbox like below and click the **Update** button to update the multiple Syncfusion NuGet packages in your application.



Upgrade NuGet packages through Package Manager Console

The **Package Manager Console** saves NuGet packages upgrade time since you don't have to search for the package you want to update, and you can just type the command to update the appropriate Syncfusion ASP.NET MVC JS2 NuGet package. Follow the steps below to upgrade the installed Syncfusion NuGet packages using the Package Manager Console in your ASP.NET MVC application.

1. To show the Package Manager Console, open your ASP.NET MVC application in Visual Studio and navigate to **Tools** in the Visual Studio menu and after hovering **NuGet Package Manager**, select **Package Manager Console**.



2. The Package Manager Console will be shown at the bottom of the screen. You can install the Syncfusion ASP.NET MVC JS2 NuGet packages by enter the following NuGet update commands.

Update specified Syncfusion ASP.NET MVC JS2 NuGet package

The below command will update the Syncfusion ASP.NET MVC JS2 NuGet package in the default ASP.NET MVC application.

Update-Package <Package Name>

For example: Update-Package Syncfusion.EJ2.MVC5

Update specified Syncfusion ASP.NET MVC JS2 NuGet package in specified ASP.NET MVC application

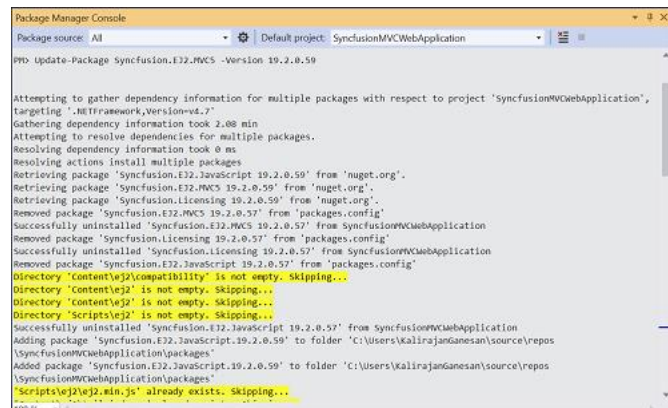
The below command will update the Syncfusion ASP.NET MVC JS2 NuGet package in the given ASP.NET MVC Web application alone.

Update-Package <Package Name> -ProjectName <Project Name>

For example: Update-Package Syncfusion.EJ2.MVC5 -ProjectName SyncfusionMVCWebApplication

3. By default, the package will be installed with latest version. You can give the required version with the -Version term like below to install the Syncfusion ASP.NET MVC JS2 NuGet packages in the appropriate version.

Update-Package Syncfusion.EJ2.MVC5 -Version 19.2.0.59



```
Package Manager Console
Package source: All
Default project: SyncfusionMVCWebApplication
PM> Update-Package Syncfusion.EJ2.MVC5 -Version 19.2.0.59

Attempting to gather dependency information for multiple packages with respect to project 'SyncfusionMVCWebApplication',
targeting '.NETFramework,Version=v4.7'.
Gathering dependency information took 2.08 min
Attempting to resolve dependencies for multiple packages.
Resolving dependency information took 0 ms
Resolving actions to install multiple packages
Retrieving package 'Syncfusion.EJ2.JavaScript 19.2.0.59' from 'nuget.org'.
Retrieving package 'Syncfusion.EJ2.MVC5 19.2.0.59' from 'nuget.org'.
Retrieving package 'Syncfusion.Licensing 19.2.0.59' from 'nuget.org'.
Removed package 'Syncfusion.EJ2.MVC5 19.2.0.57' from 'packages.config'
Successfully uninstalled 'Syncfusion.EJ2.MVC5 19.2.0.57' from SyncfusionMVCWebApplication
Removed package 'Syncfusion.Licensing 19.2.0.57' from 'packages.config'
Successfully uninstalled 'Syncfusion.Licensing 19.2.0.57' from SyncfusionMVCWebApplication
Removed package 'Syncfusion.EJ2.JavaScript 19.2.0.57' from 'packages.config'
Directory 'Content\ej2\compatibility' is not empty. Skipping...
Directory 'Content\ej2' is not empty. Skipping...
Directory 'Scripts\ej2' is not empty. Skipping...
Successfully uninstalled 'Syncfusion.EJ2.JavaScript 19.2.0.57' from SyncfusionMVCWebApplication
Adding package 'Syncfusion.EJ2.JavaScript.19.2.0.59' to folder 'c:\Users\KalirajanGanesan\source\repos
\SynfusionMVCWebApplication\packages'
Added package 'Syncfusion.EJ2.JavaScript.19.2.0.59' to folder 'c:\Users\KalirajanGanesan\source\repos
\SynfusionMVCWebApplication\packages'
'Scripts\ej2\ej2.min.js' already exists. Skipping...
Done
```

4. The Syncfusion ASP.NET MVC JS2 NuGet package, as well as its dependencies, will be updated by the NuGet package manager.

Upgrading Syncfusion ASP.NET MVC Extensions

Syncfusion publishes the Visual Studio extension in the [Visual Studio marketplace](#) for every Syncfusion Volume releases, with exciting new features and Service Pack release with major bug fixes in the volume releases.

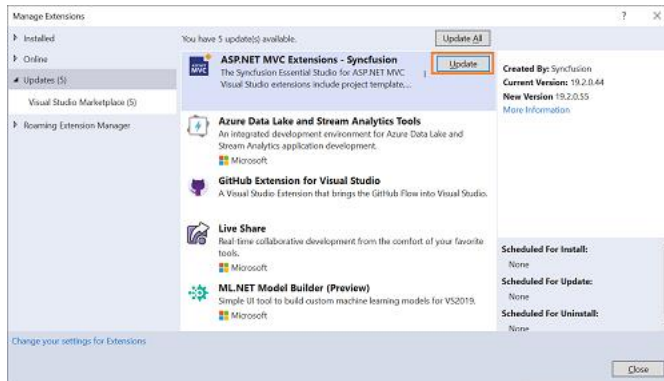
You can upgrade to our latest version from any installed Syncfusion version.

Upgrading to the latest version

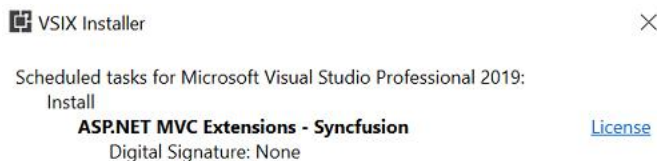
1. In Visual Studio go to Extensions -> Manage Extensions -> Updates and find the Syncfusion ASP.NET MVC extension.

Note: In Visual Studio 2017 or lower, go to Tools -> Extensions and Updates.

2. Then, click on the Update button to update the extension.



3. Now close the Visual Studio and click on the Modify button in VSIX installer dialog.



By clicking "Modify", you agree with the above license terms (if any) and the installation of any prerequisites.

Modify

Cancel

Appearance

Syncfusion ASP.NET MVC Themes

The following list of themes are included in the Syncfusion ASP.NET MVC controls library.

|Theme |Style Sheet Name|

|-----|-----|

|Bootstrap 5 | bootstrap5.css |

|Bootstrap 5 Dark | bootstrap5-dark.css |

|Bootstrap 4 | bootstrap4.css |

|Bootstrap 3 | bootstrap.css |

|Bootstrap 3 Dark | bootstrap-dark.css |

|Google's Material | material.css |

|Google's Material-Dark | material-dark.css |

|Tailwind CSS | tailwind.css |

|Tailwind CSS Dark | [tailwind-dark.css](#) |

|Fluent | [fluent.css](#) |

|Fluent Dark | [fluent-dark.css](#) |

|Microsoft Office Fabric | [fabric.css](#) |

|Microsoft Office Fabric Dark | [fabric-dark.css](#) |

|High Contrast | [highcontrast.css](#) |

The Syncfusion ASP.NET MVC Bootstrap theme is designed based on Bootstrap v3, whereas the Bootstrap4 theme is designed based on Bootstrap v4. Syncfusion ASP.NET MVC themes provide support for the Fusion Theme that can only be downloaded from [ThemeStudio](#).

Reference themes in ASP.NET MVC application

Syncfusion ASP.NET MVC themes can be used in your ASP.NET MVC application by referencing the style sheet. Refer the ASP.NET MVC Styles Sheet inside the `<head>` element of `~/Views/Shared/_Layout.cshtml` layout page.

Using the below approaches the themes can be referenced in the ASP.NET MVC application,

1. [CDN](#) - Used to reference complete css via static web assets.
2. [CRG](#) - Used to generate resources only for the selected (used) components.
3. [Theme Studio](#) - Used to customize and generate themes only for the selected (used) components.

CDN Reference

Instead of using a local resource on your server, you can use a cloud CDN to reference the theme style sheets. CDN Stands for "Content Delivery Network". A CDN is a group of servers distributed in different locations. While CDN are often used to host websites, they are commonly used to provide other types of downloadable data as well. Examples include software programs, images, videos, and streaming media.

Syncfusion ASP.NET MVC Themes are available in the CDN. Make sure that the version in the URLs matches the version of the Syncfusion Essential JS 2 ASP.NET MVC Package you are using.

~/ LAYOUT.CSHTML

```
<head>
<link href="https://cdn.syncfusion.com/ej2/{{ site.ej2version
  }}/bootstrap5.css" rel="stylesheet"/>
</head>
```

| Theme Name | CDN Reference |

|---| ---|

| Bootstrap 5 | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/bootstrap5.css> |

| Bootstrap 5 Dark | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/bootstrap5-dark.css> |

| Bootstrap 4 | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/bootstrap4.css> |

| Bootstrap 3 | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/bootstrap.css> |

| Bootstrap 3 Dark | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/bootstrap-dark.css> |

| Google's Material | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/material.css> |

| Google's Material Dark | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/material-dark.css> |

| Tailwind CSS | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/tailwind.css> |

| Tailwind Dark CSS | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/tailwind-dark.css> |

| Fluent | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/fluent.css> |

| Fluent Dark | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/fluent-dark.css> |

| Microsoft Office Fabric | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/fabric.css> |

| Microsoft Office Fabric Dark | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/fabric-dark.css> |

| High Contrast | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/highcontrast.css> |

Change theme dynamically

In the ASP.NET MVC application, the application theme can be changed dynamically by changing its style sheet reference in code.

1. Add **id** attribute in the Syncfusion style sheet references in **~/Views/Shared/_Layout.cshtml** page like below.

~/LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link id="cssfile" rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/bootstrap5.css" />
</head>
```

2. Create a **ThemeDetails.cs** model page and use the following code for dropdown data on the **~/Models/ThemeDetails.cs** page.

~/THEMEDETAILS.CS

```
using System.Collections.Generic;
namespace ThemeSwitch.Models
{
    public class ThemeDetail
    {
        public string ID { get; set; }
        public string Text { get; set; }
        public List<ThemeDetail> ThemeDetails()
        {
            List<ThemeDetail> themes = new List<ThemeDetail>();
            themes.Add(new ThemeDetail() { ID = "material", Text = "Material" });
            themes.Add(new ThemeDetail() { ID = "bootstrap", Text = "Bootstrap" });
            themes.Add(new ThemeDetail() { ID = "fabric", Text = "Fabric" });
            themes.Add(new ThemeDetail() { ID = "bootstrap4", Text = "Bootstrap 4" });
            themes.Add(new ThemeDetail() { ID = "tailwind", Text = "TailWind" });
            themes.Add(new ThemeDetail() { ID = "tailwind-dark", Text = "TailWind Dark"
            });
            themes.Add(new ThemeDetail() { ID = "material-dark", Text = "Material Dark"
            });
        }
    }
}
```



```

themes.Add(new ThemeDetail() { ID = "bootstrap-dark", Text = "Bootstrap Dark" });
themes.Add(new ThemeDetail() { ID = "fabric-dark", Text = "Fabric Dark" });
themes.Add(new ThemeDetail() { ID = "highcontrast", Text = "High Contrast" });
return themes;
}
}
}

```

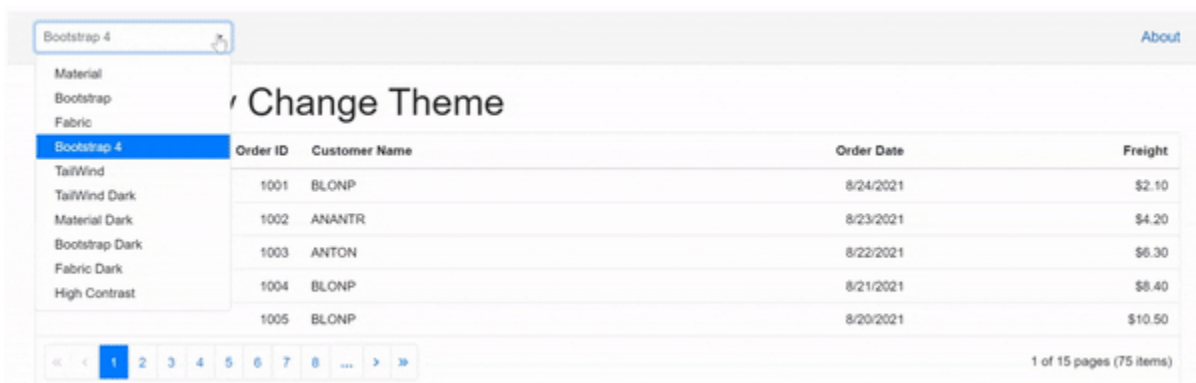
3. Implement a theme change dynamically using the Syncfusion ASP.NET MVC dropdown control in the application as in the below code.

CSHTML

```

@using Syncfusion.EJ2.DropDowns;
@using ThemeSwitch.Models;
<div>
@Html.EJS().DropDownList("theme").Placeholder("Select a theme").PopupHeight("220px").Index(2).Width("20%").DataSource(
(IEnumerable<ThemeDetail>)ViewBag.data).Change("onThemeChange").Fields(new
DropDownListFieldSettings { Text = "Text", Value = "ID" }).Render()
</div>
<script type="text/javascript">
function onThemeChange(e) {
document.getElementsByTagName('body')[0].style.display = 'none';
var themeName = e.value;
let synclink = document.getElementById('cssfile');
synclink.href = 'https://cdn.syncfusion.com/ej2/{ site.ej2version }/' +
themeName + '.css';
setTimeout(function () {
document.getElementsByTagName('body')[0].style.display = 'block'; }, 500);
}
</script>

```



Note: [View sample in GitHub](#)

Size Mode for ASP.NET MVC Controls

Syncfusion ASP.NET MVC controls support touch (bigger theme) and normal size modes. Below topics explains how to enable the same in your application.

Size mode for application

You can enable touch mode (bigger theme) for an application by adding `.e-bigger` class in the `~/Content/Site.css` file and assign to the `body` element in the `~/Views/Shared/_Layout.cshtml` page.

~/SITE.CSS

```
.e-bigger {  
font-size: x-large;  
}
```

~/ LAYOUT.CSHTML

```
<body class="e-bigger">...</body>
```

Size mode for a Control

You can enable touch mode (bigger theme) for ASP.NET MVC controls by adding `.e-bigger` class and assign to the `div` which contains the control.

CSHTML

```
<div class="e-bigger">  
@Html.EJS().Calendar("calendar").Render()  
</div>  
<div class="e-bigger">  
@Html.EJS().Button("element").Content("Button").Render()  
</div>  
<div class="e-bigger">  
@Html.EJS().CheckBox("default").Label("Checked").Checked(true).Render()  
</div>  
<style>  
.e-bigger {  
font-size: x-large;  
}  
</style>
```

Change size mode for application at runtime

You can change the size mode of an application between touch and normal (mouse) mode at runtime by adding and removing the `.e-bigger` class.

Follow the below steps to change the size mode for an application at runtime.

1.Add the `e-bigger` CSS class in the `~/Content/Site.css` file.

~/SITE.CSS

```
.e-bigger {  
font-size: x-large;  
}
```

2.Refer to the following code for adding control and the JavaScript click action inside the script tag of `~/Views/Home/Index.cshtml` file to switch between touch and mouse mode using `e-bigger` class.

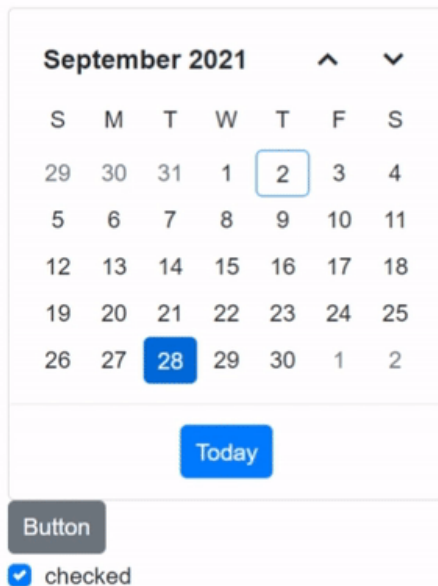
CSHTML

```
<p> Size-modes for application </p>
<p> This demo shows the Size-Modes applied for an entire application </p>
<button id="touch">Touch Mode</button>
<button id="mouse">Mouse Mode</button>
<div>
@Html.EJS().Calendar("calendar").Render()
</div>
<div>
@Html.EJS().Button("element").Content("Button").Render()
</div>
<div>
@Html.EJS().CheckBox("default").Label("Checked").Checked(true).Render()
</div>
<script>
document.getElementById("touch").addEventListener("click", function () {
document.body.classList.add('e-bigger');
});
document.getElementById("mouse").addEventListener("click", function () {
document.body.classList.remove('e-bigger');
});
</script>
```

Size-modes for application

This demo shows the Size-Modes applied for an entire application

Touch Mode Mouse Mode



Note: [View sample in GitHub](#)

Change size mode for a control at runtime

You can change the size mode of a control between touch and normal (mouse) mode at runtime by setting `.e-bigger` CSS class.

Refer to the following code, in which the `e-bigger` class is added for enabling touch mode using the for loop in ASP.NET MVC application.

CSHTML

```
<p> Size-modes for application </p>
<p> This demo shows the Size-Modes applied for Syncfusion Control </p>
<button id="touch">Touch Mode</button>
<button id="mouse">Mouse Mode</button>
<div class="control">
@Html.EJS().Calendar("calendar").Render()
</div>
<div class="control">
@Html.EJS().Button("element").Content("Button").Render()
</div>
```

```
<div class="control">
@Html.EJS().CheckBox("default").Label("Checked").Checked(true).Render()
</div>
<style>
.e-bigger {
font-size: x-large;
}
</style>
<script>
document.getElementById("touch").addEventListener("click", function () {
var controls = document.querySelectorAll('.control');
for (var index = 0; index < controls.length; index++) {
controls[index].classList.add('e-bigger');
}
});
document.getElementById("mouse").addEventListener("click", function () {
var controls = document.querySelectorAll('.control');
for (var index = 0; index < controls.length; index++) {
controls[index].classList.remove('e-bigger');
}
});
</script>
```

Size-modes for application

This demo shows the Size-Modes applied for Syncfusion Control

Touch Mode

Mouse Mode

September 2021

^

v

S	M	T	W	T	F	S
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2

Today

Button

☒ checked

Note: [View sample in GitHub](#)

See Also

Refer below topics to learn about responsiveness controls based on the available size in Syncfusion ASP.NET MVC Controls.

- [Sidebar Responsiveness](#)
- [DataGrid Responsiveness](#)

- [TreeGrid Responsiveness](#)
- [Dashboard Layout Responsiveness](#)
- [Kanban Responsiveness](#)
- [Toolbar Responsiveness](#)
- [Tab Responsiveness](#)

Overview in ASP.NET MVC Controls

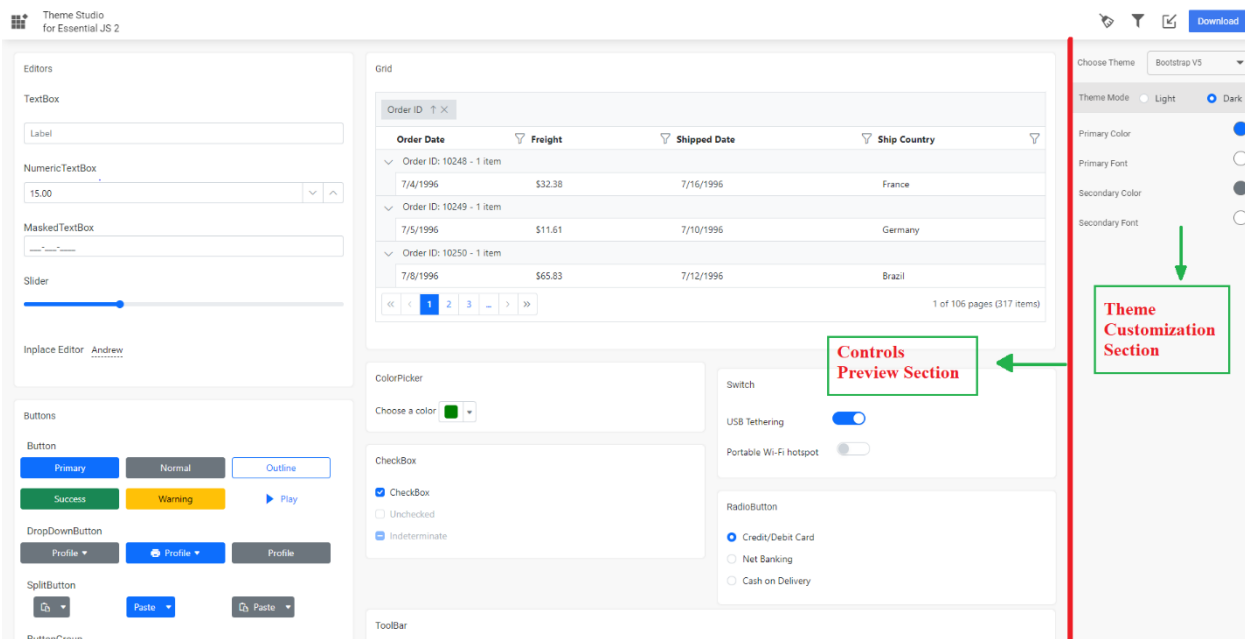
Theme Studio for Essential JS 2 can be used to customize a new theme from an existing theme. It doesn't support with Data visualization controls like Chart, Diagram, Gauge, Range Navigator, Maps.

Customizing theme color from theme studio

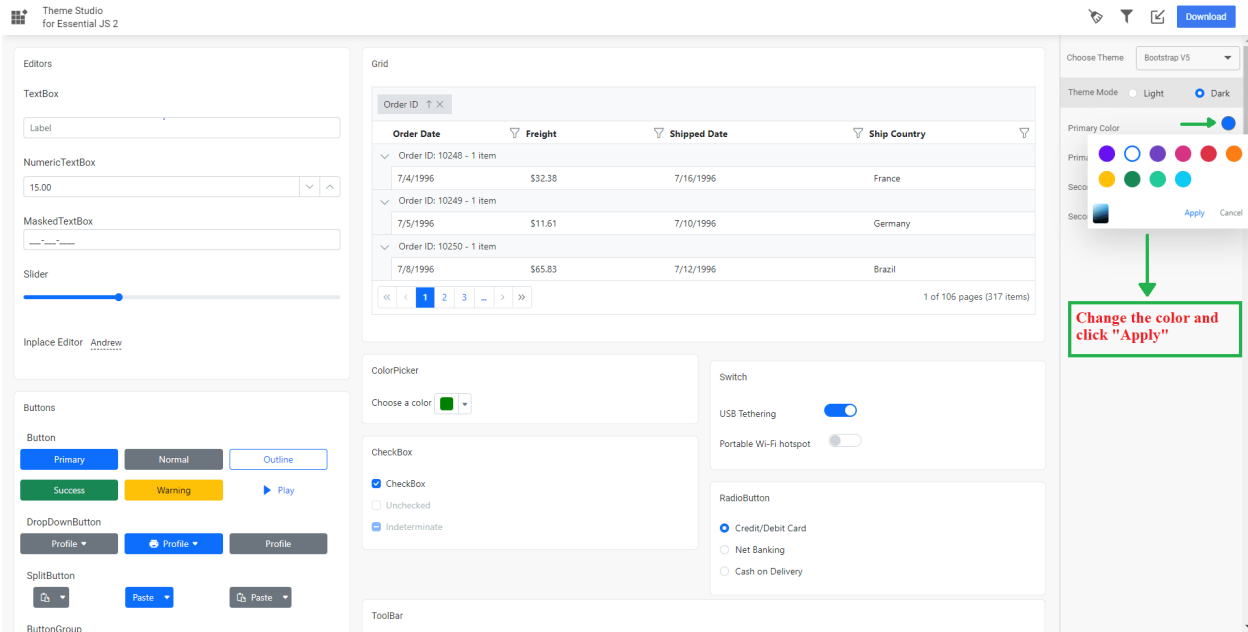
The Essential JS 2 themes are developed under the SCSS environment. Each theme has a unique common variable list. When you change the common variable color code value, it will reflect in all the Syncfusion ASP.NET MVC controls. All the Syncfusion ASP.NET MVC control styles are derived from these [theme-based common variables](#). This common variable list is handled inside the theme studio application for customizing theme-based colors.

Step 1: Open the [Syncfusion Theme Studio](#) application.

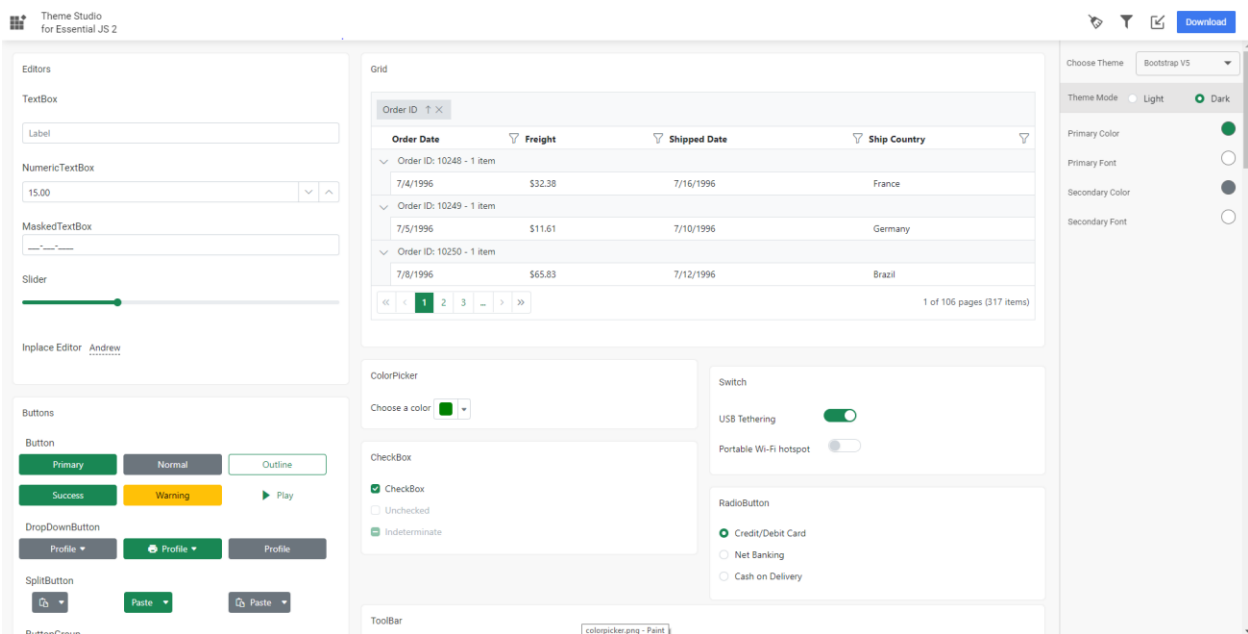
Step 2: The theme studio application page can be divided into two sections: the controls preview section on the left, and the theme customization section on the right.



Step 3: Click the color pickers in the theme customization section to select your desired colors.



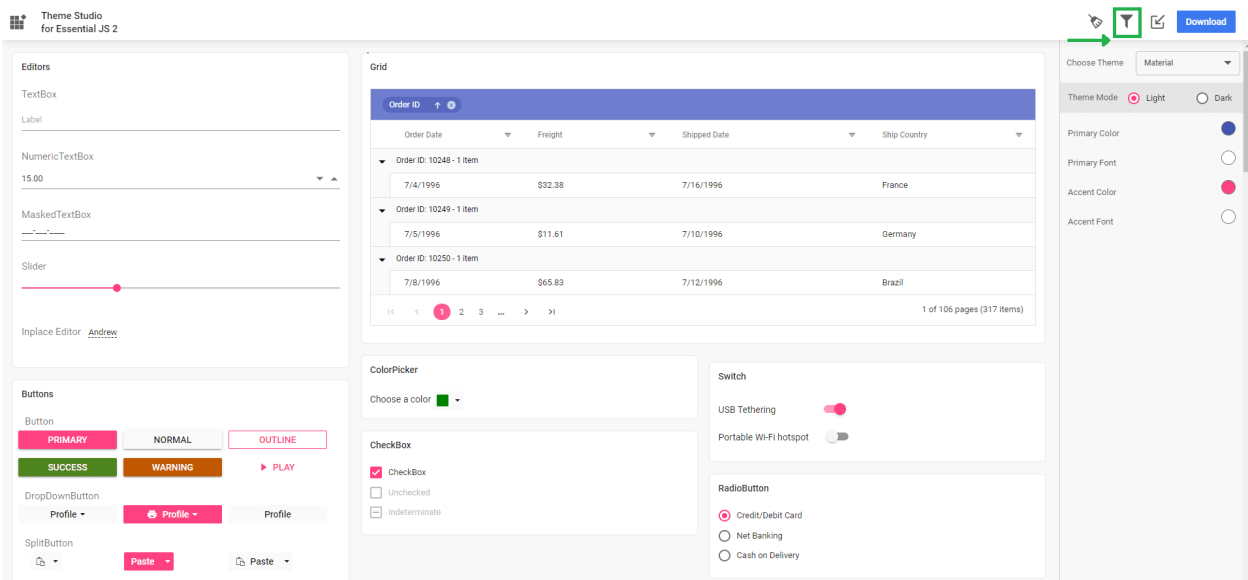
Step 4: After selecting colors with the color pickers, the Syncfusion ASP.NET MVC controls will have the newly selected colors applied to them in the controls preview section.



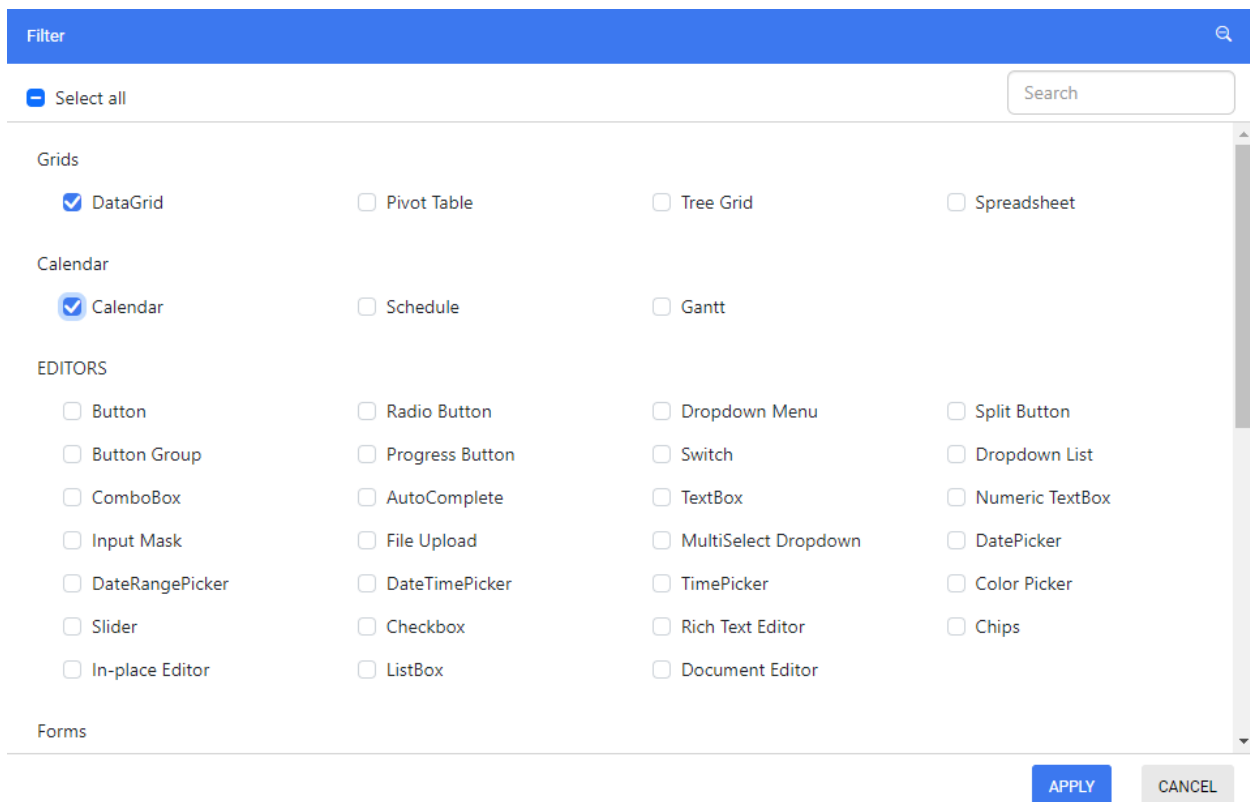
Filtering a specific list of controls

Using the theme studio, you can apply custom themes to a list of specific controls. This option is useful when you have integrated a selective list of Syncfusion ASP.NET MVC controls in your application. The theme studio will filter the selected controls and customize the final output for those controls' styles alone, reducing the final output file size.

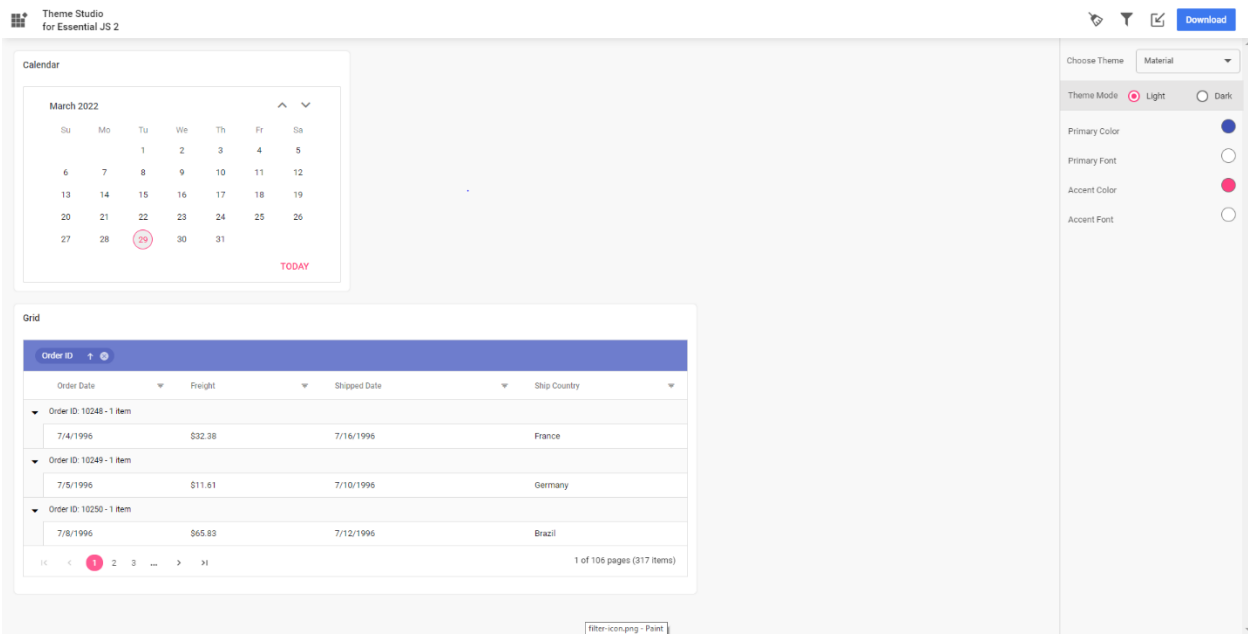
Step 1: Click the Filter icon in the top right corner and select the controls whose theme you want to customize.



Step 2: Click the Apply button in the Filter dialog. Now, only the selected controls will be rendered in the controls preview section.



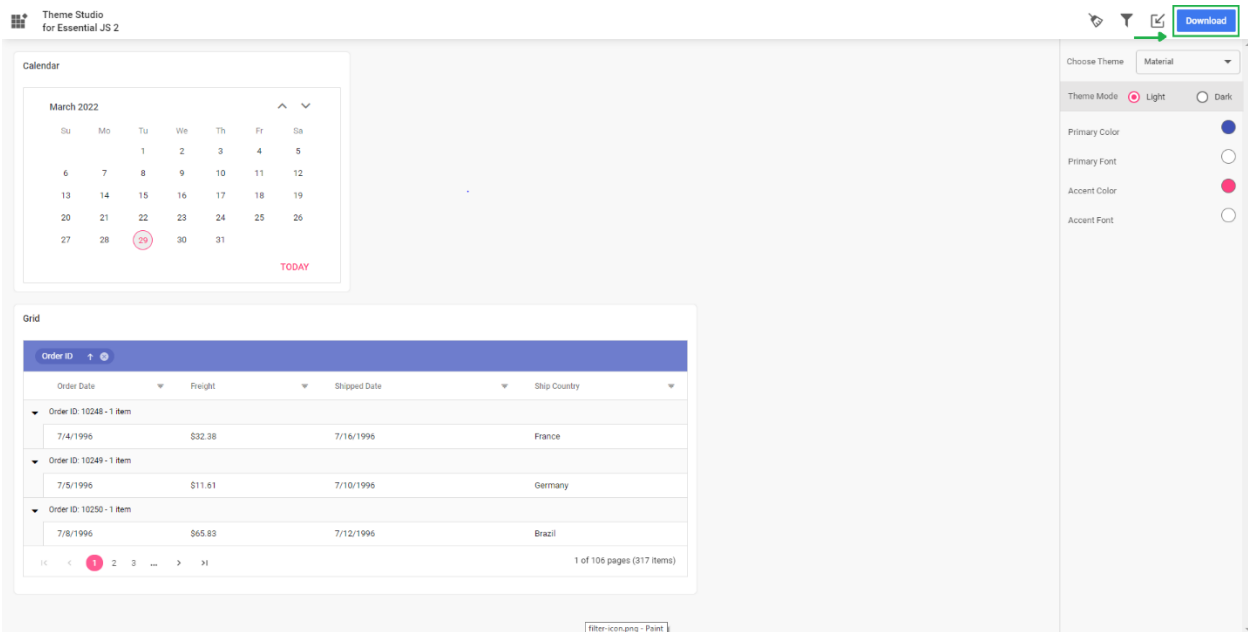
Step 3: Now you can customize the colors in the theme customization section for the controls you selected.



Download the customized theme

You can download the custom styles after customizing the theme colors.

Step 1: Click the Download button in the top right corner. The Download dialog will open.



Step 2: Assign a theme name in the File Name field and click the Download button. If your application uses both Essential JS 1 and Essential JS 2 controls, then select the Include compatibility css check box before downloading the theme. This option will generate the custom theme for Essential JS 2 compatibility styles, which are compatible as Essential JS 1 styles. Refer this [link](#) for more details about Essential JS 1 and Essential JS 2 compatibility.

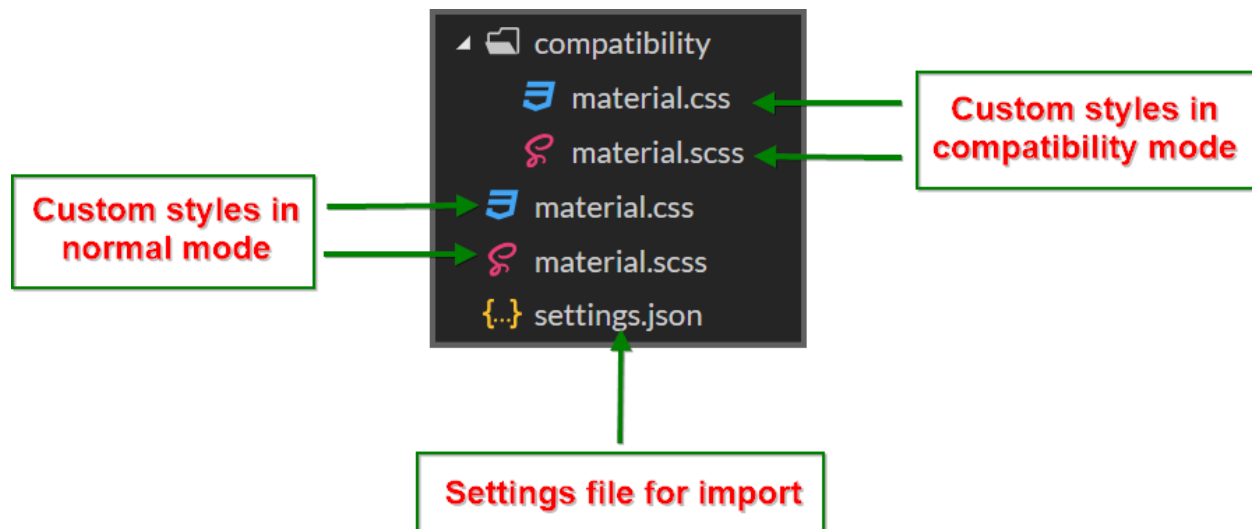
Download [X]

File Name
material

☐ Include compatibility css ⓘ

DOWNLOAD CANCEL

Step 3: The download styles will come as a zip file that contains SCSS and CSS files for the selected Syncfusion ASP.NET MVC controls. The current settings are stored in the `settings.json` file.



Using customized theme in a web application

You can directly use the customized CSS file in the web application.

Step 1: Copy/paste the customized CSS file from the download folder into your application at any folder.
Example: `styles/{file-name}.css`.

Step 2: Refer the customized CSS file reference in the `index.html` or `shared/_layout.cshtml` main page head section.

~/ _LAYOUT.CSHTML

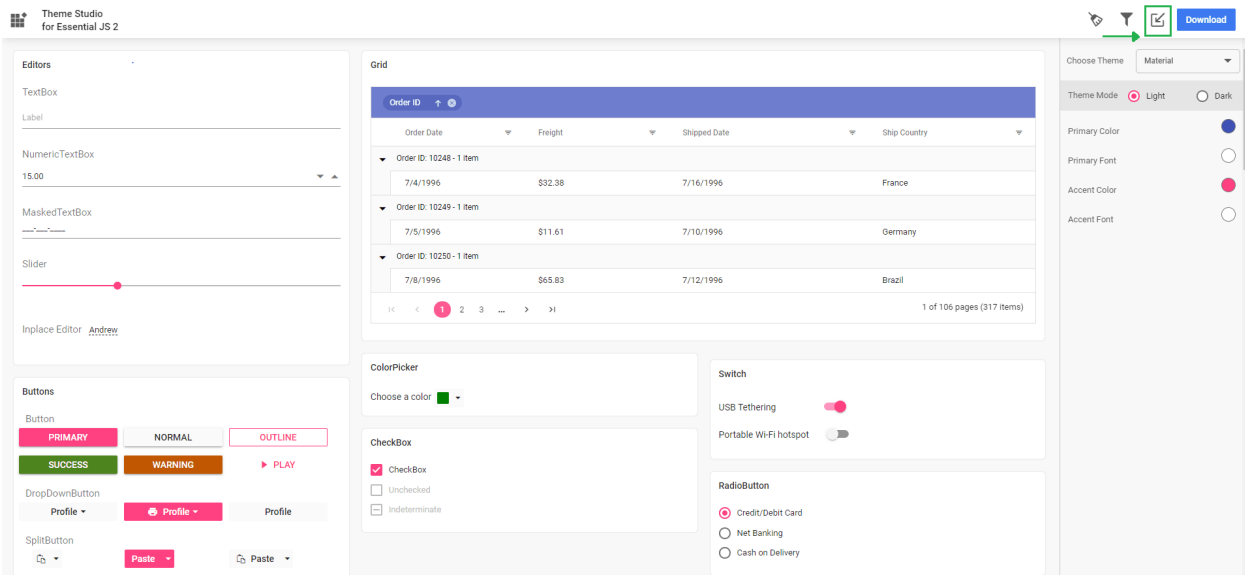
```
<head>
<link href="styles/{file-name}.css" rel="stylesheet"/>
</head>
```

Note: If you are using Essential JS 1 and Essential JS 2 controls in a same web application, then you have to copy/paste the customized CSS file from the `compatibility` folder in the download location.

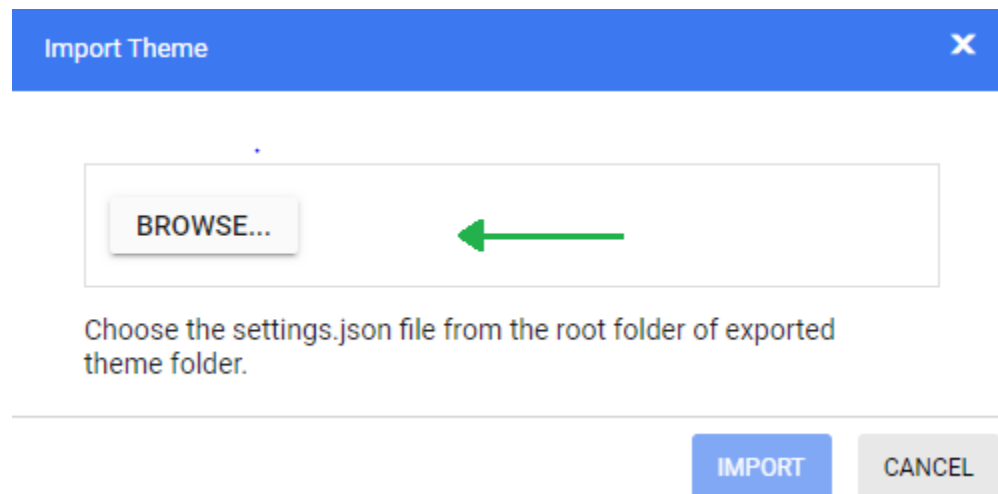
Import previously changed settings into the theme studio

When you want to change your application theme and UI design in the future, you won't need to customize the Syncfusion ASP.NET MVC controls from scratch in the theme studio. Just import the old `settings.json` file to review and update your stored settings in the theme studio application.

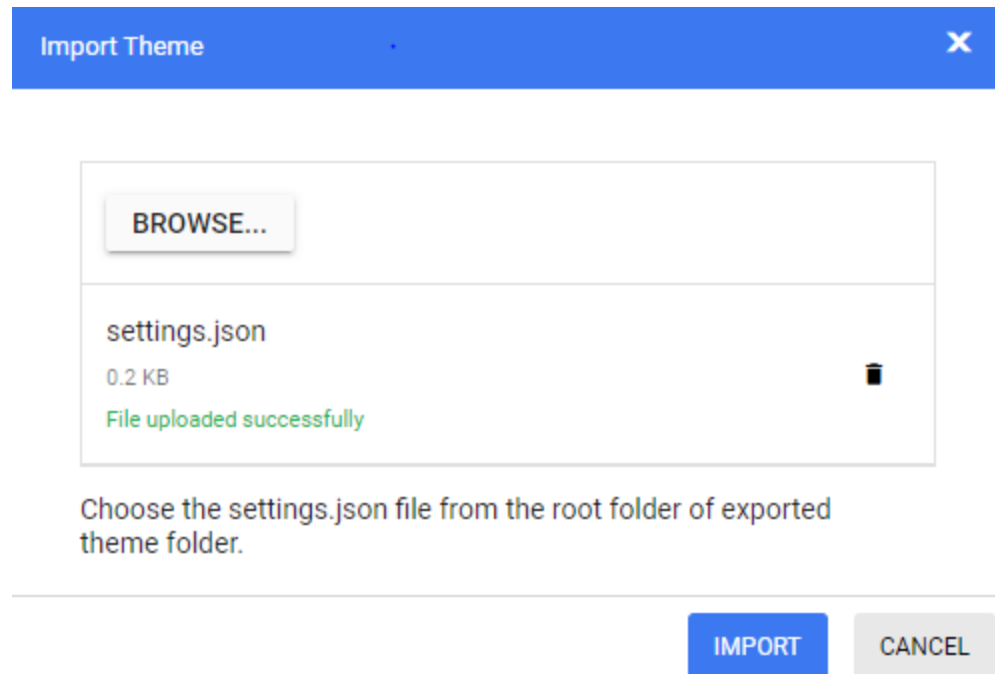
Step 1: Click the Import icon in the top right corner.



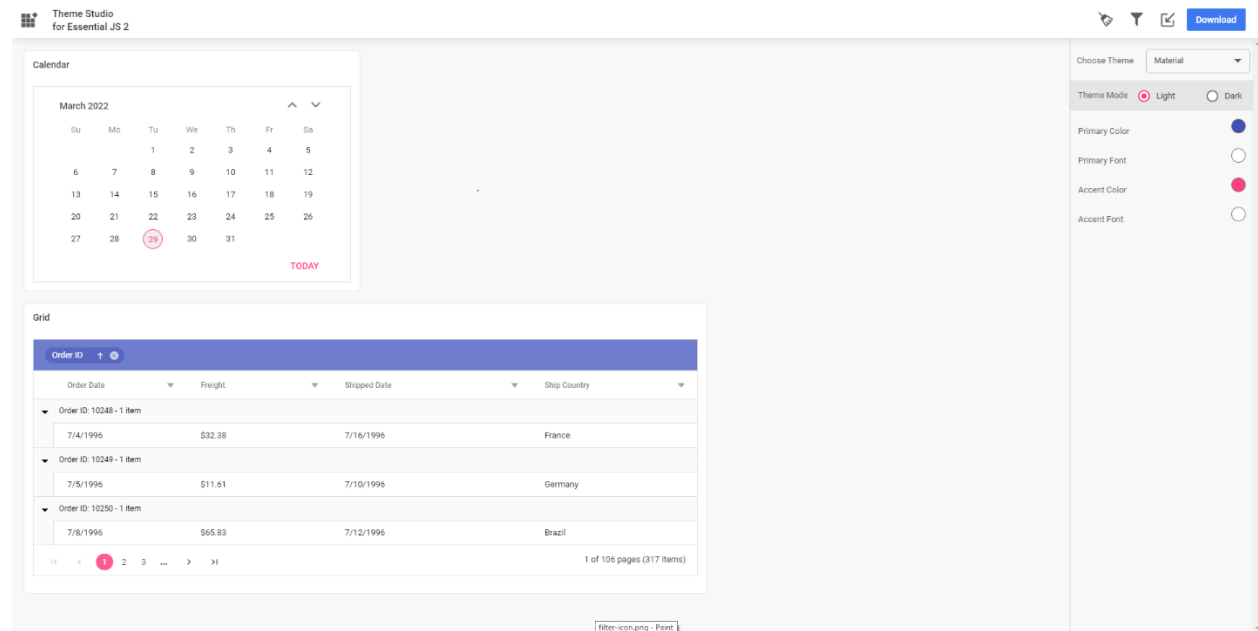
Step 2: The Import Theme dialog will open. Click the Browse button and select a `settings.json` file you exported previously.



Step 3: Click the Import button.



Step 4: The stored data will be reflected in the theme studio application. Now you can change the theme colors based on your latest design and export the theme again.



Step 5: The exported file will contain your latest changes. You can just replace the older custom style with the newer one to refresh your application.

Common Variables

The following list of common variables is used in the Syncfusion ASP.NET MVC library themes for all UI controls. You can change these variables to customize the corresponding theme.

Bootstrap 5 Theme

Name	Value (Default Theme)	Value (Dark Theme)
\$black	#000	#000
\$white	#fff	#fff
\$gray-100	#f8f9fa	#f8f9fa
\$gray-200	#e9ecef	#e9ecef
\$gray-300	#dee2e6	#dee2e6
\$gray-400	#ced4da	#ced4da
\$gray-500	#adb5bd	#adb5bd
\$gray-600	#6c757d	#6c757d
\$gray-700	#495057	#495057
\$gray-800	#343a40	#343a40
\$gray-900	#212529	#212529
\$blue	#0d6efd	#0d6efd
\$indigo	#6610f2	#6610f2
\$purple	#6f42c1	#6f42c1
\$pink	#d63384	#d63384
\$red	#dc3545	#dc3545
\$orange	#fd7e14	#fd7e14
\$yellow	#ffc107	#ffc107
\$green	#198754	#198754
\$teal	#20c997	#20c997
\$cyan	#0dcaf0	#0dcaf0

Fluent Theme

Name	Value (Default Theme)	Value (Dark Theme)
\$black	#000	#000
\$white	#fff	#fff
\$gray220	#11100f	#11100f
\$gray210	#161514	#161514
\$gray200	#1b1a19	#1b1a19
\$gray190	#201f1e	#201f1e
\$gray180	#252423	#252423
\$gray170	#292827	#292827
\$gray160	#323130	#323130
\$gray150	#3b3a39	#3b3a39
\$gray140	#484644	#484644
\$gray130	#605e5c	#605e5c
\$gray120	#797775	#797775
\$gray110	#8a8886	#8a8886
\$gray100	#979593	#979593
\$gray90	#a19f9d	#a19f9d
\$gray80	#b3b0ad	#b3b0ad
\$gray70	#bebbb8	#bebbb8
\$gray60	#c8c6c4	#c8c6c4
\$gray50	#d2d0ce	#d2d0ce
\$gray40	#e1dfdd	#e1dfdd

Name	Value (Default Theme)	Value (Dark Theme)
\$gray30	#edebe9	#edebe9
\$gray20	#f3f2f1	#f3f2f1
\$gray10	#faf9f8	#faf9f8
\$cyanblue10	#0078d4	#0078d4
\$red10	#d13438	#d13438
\$orange20	#ca5010	#ca5010
\$green20	#0b6a0b	#0b6a0b
\$cyan20	#038387	#038387

Bootstrap 4 Theme

Name	Value
\$white	#fff
\$gray-100	#f8f9fa
\$gray-200	#e9ecef
\$gray-300	#dee2e6
\$gray-400	#ced4da
\$gray-500	#adb5bd
\$gray-600	#6c757d
\$gray-700	#495057
\$gray-800	#343a40
\$gray-900	#212529
\$black	#000
\$blue	#007bff

Name	Value
\$indigo	#6610f2
\$purple	#6f42c1
\$pink	#e83e8c
\$red	#dc3545
\$orange	#fd7e14
\$yellow	#ffc107
\$green	#28a745
\$teal	#20c997
\$cyan	#17a2b8

Bootstrap Theme

Design based on bootstrap 3 theme.

Name	Value (Default Theme)	Value (Dark Theme)
\$brand-primary	#317ab9	#0070f0
\$brand-primary-darken-10	#3071a9	darken(\$brand-primary, 10%)
\$brand-primary-darken-15	#2a6496	darken(\$brand-primary, 20%)
\$brand-primary-darken-25	#1f496e	darken(\$brand-primary, 30%)
\$brand-primary-darken-35	#142f46	darken(\$brand-primary, 40%)
\$brand-primary-font	#fff	#fff
\$gray-base	#000	#1a1a1a
\$gray-darker	#222	#131313
\$gray-dark	#333	#2a2a2a
\$gray	#555	#313131

Name	Value (Default Theme)	Value (Dark Theme)
\$gray-light	#777	#393939
\$grey-44	#444	#414141
\$grey-88	#888	#484848
\$grey-99	#999	#505050
\$grey-8c	#8c8c8c	#585858
\$grey-ad	#adadad	#676767
\$grey-dark-font	#fff	#f0f0f0
\$grey-white	#fff	#6e6e6e
\$grey-lighter	#eee	#767676
\$grey-f9	#f9f9f9	#7e7e7e
\$grey-f8	#f8f8f8	#858585
\$grey-f5	#f5f5f5	#8d8d8d
\$grey-e6	#e6e6e6	#959595
\$grey-dd	#ddd	#9c9c9c
\$grey-d4	#d4d4d4	#a4a4a4
\$grey-cc	#ccc	#acacac
\$grey-light-font	#333	#fff
\$brand-success	#5cb85c	#48b14c
\$brand-success-dark	#3c763d	#358238
\$brand-info	#5bc0de	#2aaac0
\$brand-info-dark	#31708f	#208090
\$brand-warning	#f0ad4e	#fac168

Name	Value (Default Theme)	Value (Dark Theme)
\$brand-warning-dark	#8a6d3b	#f9ad37
\$brand-danger	#d9534f	#d44f4f
\$brand-danger-dark	#a94442	#c12f2f
\$brand-success-light	#dff0d8	#dff0d8
\$brand-info-light	#d9edf7	#d9edf7
\$brand-warning-light	#fcf8e3	#fcf8e3
\$brand-danger-light	#f2dede	#f2dede
\$input-border-focus	#66afe9	#104888
\$brand-success-font	#3c763d	#2f7432
\$brand-info-font	#31708f	#1a6c7a
\$brand-warning-font	#8a6d3b	#9d6106
\$brand-danger-font	#a94442	#ac2a2a
\$base-font	#000	#000
\$brand-primary-lighten-10		lighten(\$brand-primary, 10%)
\$brand-primary-lighten-15		lighten(\$brand-primary, 15%)
\$brand-primary-lighten-20		lighten(\$brand-primary, 20%)
\$brand-primary-lighten-30		lighten(\$brand-primary, 30%)
\$brand-primary-lighten-40		lighten(\$brand-primary, 40%)

Material Theme

Name	Value (Default Theme)	Value (Dark Theme)
\$accent	#e3165b	#ff80ab
\$accent-font	#fff	#000

Name	Value (Default Theme)	Value (Dark Theme)
\$primary	#3f51b5	#3f51b5
\$primary-50	#e8eaf6	#e8eaf6
\$primary-100	#c5cae9	#c5cae9
\$primary-200	#9fa8da	#9fa8da
\$primary-300	#7986cb	#7986cb
\$primary-font	#fff	#fff
\$primary-50-font	#000	#000
\$primary-100-font	#000	#000
\$primary-200-font	#000	#000
\$primary-300-font	#fff	#fff
\$grey-white	#fff	#fff
\$grey-black	#000	#000
\$grey-50	#fafafa	#fafafa
\$grey-100	#f5f5f5	#f5f5f5
\$grey-200	#eee	#eee
\$grey-300	#e0e0e0	#e0e0e0
\$grey-400	#bdbdbd	#bdbdbd
\$grey-500	#9e9e9e	#9e9e9e
\$grey-600	#757575	#757575
\$grey-700	#616161	#616161
\$grey-800	#424242	#424242
\$grey-900	#212121	#212121

Name	Value (Default Theme)	Value (Dark Theme)
\$grey-dark	#303030	#303030
\$grey-light-font	#000	#000
\$grey-dark-font	#fff	#fff
\$base-font	#000	#000
\$error-font	#f44336	#ff6652
\$success-bg		#4caf50
\$error-bg		#ff6652
\$warning-bg		#ff9800
\$info-bg		#03a9f4
\$message-font		#fff
\$success-font		#4caf50
\$warning-font		#ff9800
\$info-font		#03a9f4

Tailwind CSS Theme

Name	Value (Default Theme)	Value (Dark Theme)
\$black	#000	#000
\$white	#fff	#fff
\$transparent	transparent	transparent
\$cool-gray-50	#f9fafb	#f9fafb
\$cool-gray-100	#f3f4f6	#f3f4f6
\$cool-gray-200	#e5e7eb	#e5e7eb
\$cool-gray-300	#d1d5db	#d1d5db

Name	Value (Default Theme)	Value (Dark Theme)
\$cool-gray-400	#9ca3af	#9ca3af
\$cool-gray-500	#6b7280	#6b7280
\$cool-gray-600	#4b5563	#4b5563
\$cool-gray-700	#374151	#374151
\$cool-gray-800	#1f2937	#1f2937
\$cool-gray-900	#111827	#111827
\$red-100	#fee2e2	#fee2e2
\$red-400	#f87171	#f87171
\$red-500	#ef4444	#ef4444
\$red-600	#dc2626	#dc2626
\$red-800	#991b1b	#991b1b
\$green-100	#dcfce7	#dcfce7
\$green-500	#22c55e	#22c55e
\$green-600	#16a34a	#16a34a
\$green-700	#15803d	#15803d
\$orange-100	#ffedd5	#ffedd5
\$orange-500	#f97316	#f97316
\$orange-600	#ea580c	#ea580c
\$orange-700	#c2410c	#c2410c
\$orange-800	#9a3412	#9a3412
\$cyan-300	#67e8f9	#67e8f9
\$cyan-400	#22d3ee	#22d3ee

Name	Value (Default Theme)	Value (Dark Theme)
\$cyan-500	#06b6d4	#06b6d4
\$cyan-600	#0891b2	#0891b2
\$cyan-800	#155e75	#155e75
\$indigo-50	#eef2ff	
\$indigo-100	#e0e7ff	
\$indigo-200	#c7d2fe	
\$indigo-300	#a5b4fc	
\$indigo-400	#818cf8	
\$indigo-500	#6366f1	
\$indigo-600	#4f46e5	
\$indigo-700	#4338ca	
\$indigo-800	#3730a3	
\$indigo-900	#312e81	
\$green-400		#4ade80
\$light-blue-50		#f0f9ff
\$light-blue-100		#e0f2fe
\$light-blue-400		#38bdf8
\$light-blue-500		#0ea5e9
\$light-blue-600		#0284c7
\$light-blue-700		#0369a1
\$light-blue-800		#075985

Microsoft Office Fabric Theme

Name	Value (Default Theme)	Value (Dark Theme)
\$theme-primary	#0078d6	#0074cc
\$theme-dark-alt	darken(\$theme-primary, 3%)	darken(\$theme-primary, 3%)
\$theme-dark	darken(\$theme-primary, 10%)	darken(\$theme-primary, 6%)
\$theme-darker	darken(\$theme-primary, 18%)	darken(\$theme-primary, 10%)
\$theme-secondary	lighten(\$theme-primary, 3%)	lighten(\$theme-primary, 3%)
\$theme-tertiary	lighten(\$theme-primary, 21%)	lighten(\$theme-primary, 21%)
\$theme-light	lighten(\$theme-primary, 44%)	lighten(\$theme-primary, 44%)
\$theme-lighter	lighten(\$theme-primary, 49%)	lighten(\$theme-primary, 49%)
\$theme-lighter-alt	lighten(\$theme-primary, 55%)	lighten(\$theme-primary, 55%)
\$neutral-white	#fff	#201f1f
\$neutral-lighter-alt	#f8f8f8	#282727
\$neutral-lighter	#f4f4f4	#333232
\$neutral-light	#eaeaea	#414040
\$neutral-quintenaryalt	#dadada	#4a4848
\$neutral-quintenary	#d0d0d0	#514f4f
\$neutral-tertiary-alt	#c8c8c8	#6f6c6c
\$neutral-tertiary	#a6a6a6	#9a9a9a
\$neutral-secondary-alt	#767676	#c8c8c8
\$neutral-secondary	#666	#dadada
\$neutral-primary	#333	#fff
\$neutral-dark	#212121	#f4f4f4

Name	Value (Default Theme)	Value (Dark Theme)
\$neutral-black	#000	#f8f8f8
\$alert-bg	#deecf9	#bf7500
\$error-bg	#fde7e9	#cd2a19
\$success-bg	#dff6dd	#37844d
\$theme-dark-font	#fff	#fff
\$theme-primary-font	#fff	#fff
\$theme-light-font	#333	#000
\$neutral-light-font	#333	#dadada
\$neutral-light-fontalt	#000	#fff
\$grey-dark-font	#fff	#000
\$base-font	#333	#dadada
\$message-font	#333	#fff
\$alert-font	#d83b01	#ff9d48
\$error-font	#a80000	#ff5f5f
\$success-font	#107c10	#8eff8d
\$info-bg		#1e79cb
\$info-font		#62cfff

High Contrast Theme

Name	Value
\$selection-bg	#ffd939
\$selection-font	#000
\$selection-border	#ffd939

Name	Value
\$hover-bg	#685708
\$hover-font	#fff
\$hover-border	#fff
\$border-default	#969696
\$border-alt	#757575
\$border-fg	#fff
\$border-fg-alt	#ffd939
\$bg-base-0	#000
\$bg-base-5	#0d0d0d
\$bg-base-10	#1a1a1a
\$bg-base-15	#262626
\$bg-base-20	#333
\$bg-base-75	#bfbfbf
\$bg-base-100	#fff
\$header-font	#ffd939
\$header-font-alt	#fff
\$content-font	#fff
\$content-font-alt	#969696
\$link	#8a8aff
\$invert-font	#000
\$success-bg	#166600
\$error-bg	#b30900

Name	Value
\$message-font	#fff
\$alert-bg	#944000
\$info-bg	#0056b3
\$success-alt	#2bc700
\$error-alt	#ff6161
\$alert-alt	#ff7d1a
\$info-alt	#66b0ff
\$disable	#757575

Icons Library

The Syncfusion Essential JS 2 provides the set of **base64** formatted font icons, that can be utilized in the ASP.NET MVC application.

Steps to Use Icon

1.Add a class **e-icons** to the HTML element that shows the icon. This class contains the font-family and common property of the font icons. 2.Add the icon class with corresponding icon content from the [available icons](#). For example, the below code snippet represents the search icon class.

CSHTML

```
<style>
.e-search:before{
content:'\e993';
}
</style>
```

3.Add **e-icons** and **e-search** class to the HTML element.

CSHTML

```
<span class="e-icons e-search"></span>
```

The below code snippet represents the complete example of icon usage.

CSHTML

```
<style>
.e-search:before{
content:'\e993';
}
.e-upload:before{
```

```

content: '\e725';
}
.e-font:before{
content: '\e34c';
}
</style>
<div class="icons">
<ul>
<li><span class="e-icons e-search"></span></li>
<li><span class="e-icons e-settings"></span></li>
<li><span class="e-icons e-upload"></span></li>
<li><span class="e-icons e-font"></span></li>
</ul>
</div>

```

Icons Size

You can change the icon size by adding `e-small`, `e-medium` and `e-large` class in the HTML element.

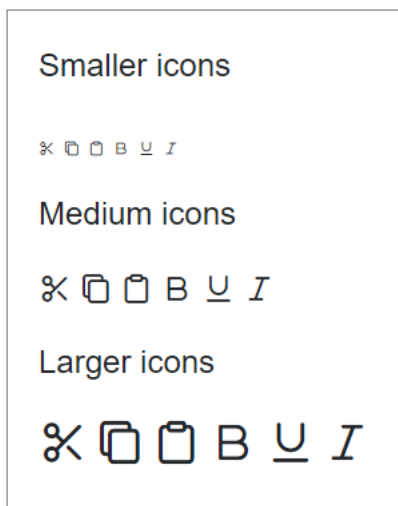
- When icon size set to Small, the font size will be 8px.
- When icon size set to Medium, the font size will be 16px.
- When icon size set to Large, the font size will be 24px.

CSHTML

```

<p>Smaller icons</p>
<span class="e-icons e-cut e-small"></span>
<span class="e-icons e-copy e-small"></span>
<span class="e-icons e-paste e-small"></span>
<span class="e-icons e-bold e-small"></span>
<span class="e-icons e-underline e-small"></span>
<span class="e-icons e-italic e-small"></span>
<p>Medium icons</p>
<span class="e-icons e-cut e-medium"></span>
<span class="e-icons e-copy e-medium"></span>
<span class="e-icons e-paste e-medium"></span>
<span class="e-icons e-bold e-medium"></span>
<span class="e-icons e-underline e-medium"></span>
<span class="e-icons e-italic e-medium"></span>
<p>Larger icons</p>
<span class="e-icons e-cut e-large"></span>
<span class="e-icons e-copy e-large"></span>
<span class="e-icons e-paste e-large"></span>
<span class="e-icons e-bold e-large"></span>
<span class="e-icons e-underline e-large"></span>
<span class="e-icons e-italic e-large"></span>

```



Tooltip for icons

`title` attribute is used to improve accessibility with screen readers and shows a tooltip on mouseover. The following example code displays tooltip text for appropriate icons.

CSHTML

```
<span class="e-icons e-upload-1" title="Upload"></span>
<span class="e-icons e-download" title="Download"></span>
<span class="e-icons e-undo" title="Undo"></span>
<span class="e-icons e-redo" title="Redo"></span>
<span class="e-icons e-align-top" title="AlignTop"></span>
<span class="e-icons e-align-bottom" title="AlignBottom"></span>
<span class="e-icons e-align-middle" title="AlignMiddle"></span>
```



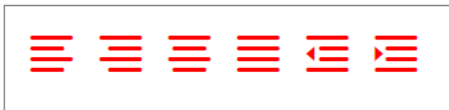
Icon appearance customization

You can customize color and size by overriding the `e-icons` class. The following example code demonstrates the custom font-size and color for icons.

CSHTML

```
<span class="e-icons e-align-top"></span>
<span class="e-icons e-align-bottom"></span>
<span class="e-icons e-align-middle"></span>
<span class="e-icons e-justify"></span>
<span class="e-icons e-decrease-indent"></span>
<span class="e-icons e-increase-indent"></span>
<style>
.e-icons{
color: #ff0000;
```

```
font-size: 26px !important;
}
</style>
```



Third party icons integration

The Syncfusion ASP.NET MVC Toolbar control supports to render custom font icons using the [prefixIcon](#) property. To render custom font icons define the required font CSS that provides the required font name, font size, and content for the icon.

The following code explains how to render **open-ionic** icons using **prefixIcon** property.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
@Html.EJS().Toolbar("defaultToolbar").Width("600px").Items(new
List<ToolbarItem> {
    new ToolbarItem { PrefixIcon = "oi oi-list-rich" },
    new ToolbarItem { PrefixIcon = "oi oi-account-login" },
    new ToolbarItem { PrefixIcon = "oi oi-account-logout" },
    new ToolbarItem { PrefixIcon = "oi oi-action-redo" },
    new ToolbarItem { PrefixIcon = "oi oi-action-undo" },
    new ToolbarItem { PrefixIcon = "oi oi-clock" },
    new ToolbarItem { PrefixIcon = "oi oi-audio" },
    new ToolbarItem { PrefixIcon = "oi oi-bluetooth" }
}).Render()
```



HTML attribute support

You can add the additional HTML attributes to the Syncfusion ASP.NET MVC Button using [HtmlAttributes](#) property. All syncfusion ASP.NET MVC controls support HTML attribute.

The following example shows the font size customization using **HtmlAttributes** directive.

CSHTML

```
@{IDictionary<string, object> customAttribute = new Dictionary<string,
object>()
{
    { "style", "font-size: 20px" }
}}
```

```
};
}
@Html.EJS().Button("button").IconCss("e-icons e-copy").HtmlAttributes(customAttribute).Render()
```

Using icons directly in HTML element

The built-in Syncfusion icons can be rendered directly in the HTML element by defining **e-icons** class that contains the font-family and common property of font icons, and defining the [available icon's](#) class with **e-** prefix.

The following code example explains the direct rendering of Syncfusion **download** icon in the span element.

CSHTML

```
<span class="e-icons e-download"></span>
```

Icons list

The complete pack of Syncfusion ASP.NET MVC icons is listed in the following table. The corresponding icon content can be referred to the content section.

<!-- markdownlint-disable MD033 -->

Bootstrap 5

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/bootstrap5/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Bootstrap 4

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/bootstrap4/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Bootstrap

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/bootstrap/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Material

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/material/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Tailwind CSS

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/tailwind/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Office Fabric

```
<iframe class="doc-sample-frame" src="https://ej2.syncfusion.com/products/icons/fabric/demo.html"
style="height:1000px;width:100%;"></iframe>
```

High Contrast

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/highcontrast/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Common

Reference Scripts in ASP.NET MVC Application

This section provides information about reference scripts from CDN and Custom resource generator (CRG) for Syncfusion ASP.NET MVC controls.

CDN Reference

Syncfusion hosts every ASP.NET MVC control as a separate node package in CDN, from which scripts and style sheets of the individual package can be loaded. Syncfusion also hosts a single node package with all ASP.NET MVC controls on it, from which scripts and style sheets of all controls can be loaded as single script and style file.

Here, the Syncfusion ASP.NET MVC CDN URL for the both Individual ASP.NET MVC control package and Complete ASP.NET MVC controls package has been explained.

Warning: The un-versioned CDN links which always maintains latest version scripts are deprecated from 2022 Vol1 - 20.1 version. These links are available with 19.4 version scripts to avoid breaking in sites and apps that uses un-versioned links.

CDN Reference for all controls

The primary goal of all the ASP.NET MVC controls package is to help the novice to get started with Syncfusion ASP.NET MVC control by referring the single line for script and styles without bothering about the dependency graph of the ASP.NET MVC controls.

| controls | CDN Reference |

| --- | --- |

| Scripts reference for all controls| <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/dist/ej2.min.js> |

| Styles reference for all controls | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/{THEME-NAME}.css> |

Note: Syncfusion will never recommend all controls CDN for real-time projects. Because, the size of this CDN impacts website/app loading time since this package includes all the Syncfusion ASP.NET MVC controls.

Add the EJ2 CDN client-side resources to the `<head>` element of the `~/Views/Shared/_Layout.cshtml` layout page.

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/material.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Individual control CDN Reference

The primary goal of individual control CDN is to optimize the loading time and memory of the website/app in the production stage. The order of individual control package loading should be in line with its dependency graph. The CDN of the Dependency Packages should be included manually before the intended individual control package CDN.

| controls | CDN Reference |

| --- | --- |

| Scripts reference for individual control | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/{PACKAGE-NAME}/dist/global/{PACKAGE-NAME}.min.js> |

| Styles reference for individual control | <https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/{PACKAGE-NAME}/styles/{THEME-NAME}.css> |

Add the CDN client-side resources in the `<head>` element of the `~/Views/Shared/_Layout.cshtml` layout page.

For example, the scripts and styles for the ASP.NET MVC Calendar control are listed below.

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/ej2-calendars/styles/material.css" />
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/ej2-base/styles/material.css" />
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/ej2-buttons/styles/material.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/ej2-
base/dist/global/ej2-base.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version }}/ej2-
calendars/dist/global/ej2-calendars.min.js"></script>
</head>
```

In addition to above, Syncfusion ASP.NET MVC controls provides latest scripts and styles in CDN without versioning. You can use this in development environment if you want to always use the latest version of scripts and styles. It is not recommended to use this in production environment.

| controls | CDN Reference |

| --- | --- |

| Scripts reference for all controls | <https://cdn.syncfusion.com/ej2/dist/ej2.min.js> |

| Styles reference for all controls | <https://cdn.syncfusion.com/ej2/{THEME-NAME}.css> |

Node Package Manager (NPM)

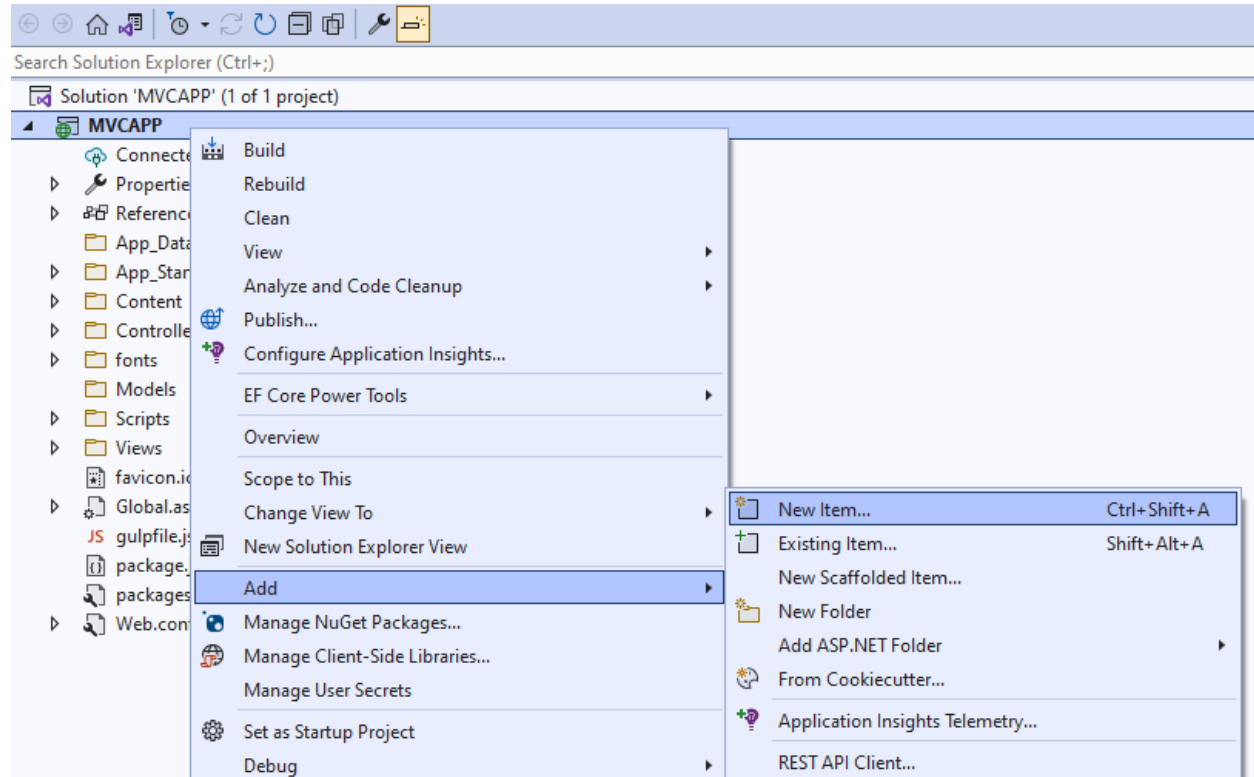
A package in Node.js consists of a set of files needed for a JavaScript module which can be included in any web application. NPM is the popular package manager for both public and private packages.

Syncfusion NPM packages

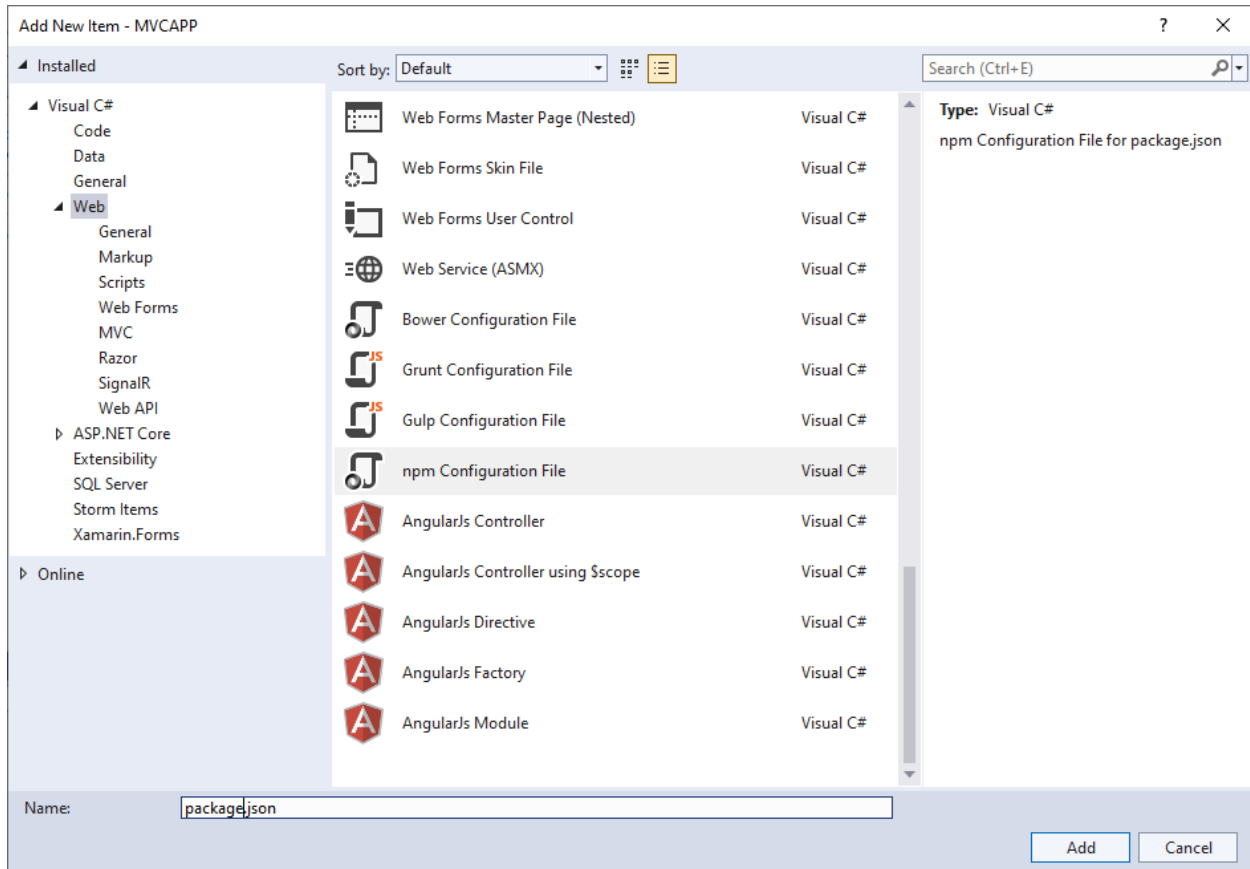
Scripts and style sheets of Syncfusion ASP.NET MVC controls can be included in an ASP.NET MVC web application using NPM packages, since Syncfusion publishes its ASP.NET MVC controls as the scoped package in NPM.

Installing NPM packages in ASP.NET MVC Web Application

1. Open the Source Explorer and right click the application name. Then, select “**Add New Item**” menu item to open the “**Add New Item**” window.



2. Select “**Web**” on the left side Tree View and select “**npm configuration File**” in “**Add New Item**” window which will include and configure “**package.json**” file in root folder of the ASP.NET MVC web application.

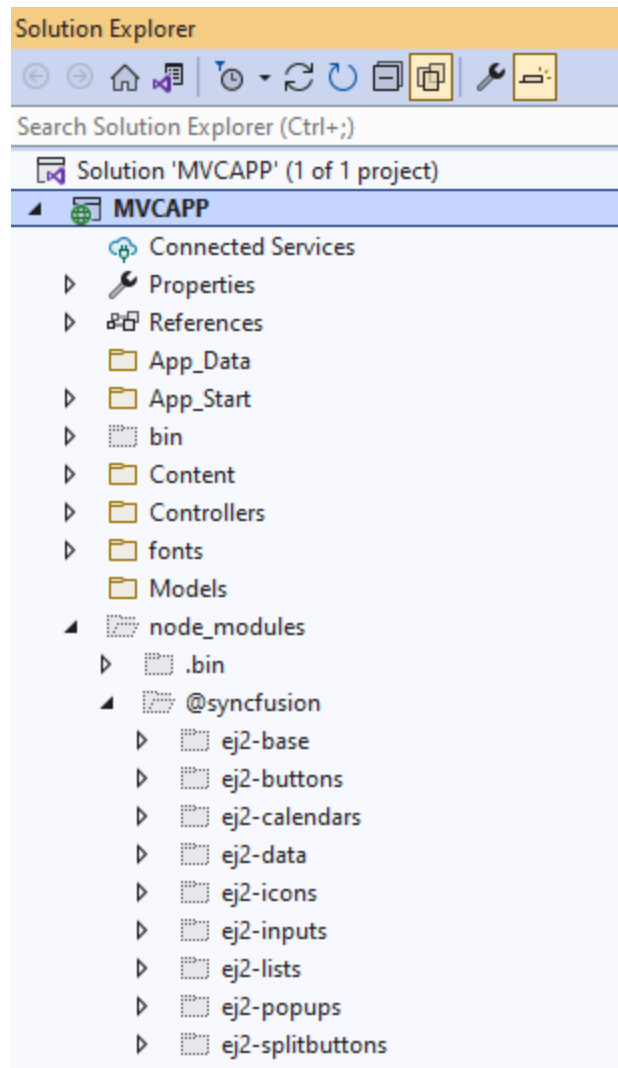


3. Open the Source Explorer and right click on the application name. Then, select **“Open Folder in File Explorer”**. 4. Open the Command Prompt for this location and install the required Syncfusion packages using **“npm install @syncfusion/{PACKAGE_NAME} --save”**

.NET CLI

```
npm install @syncfusion/ej2-calendars --save
```

5. Installed packages with all its dependencies can be found under the **“node_modules”** folder as shown in the following image.



From Installed Packages

Scripts and style sheets of Syncfusion ASP.NET MVC controls from locally-installed packages can be included in an ASP.NET MVC web application using the following two methods.

Note: Files in the **Content** folder can be accessed only in client-side, hence Syncfusion scripts and style sheets should be copied from **node_modules** to **Content**.

By using either one of the following methods, Client-Side Resource can be loaded in ASP.NET MVC web application:

1. Copying scripts and styles using gulp
2. Generating scripts and styles using [CRG \(Custom Resource Generator\)](#)

Copying by Gulp

Install required Syncfusion ASP.NET MVC control packages as mentioned in [“Installing NPM Packages in ASP.NET MVC Web Application”](#).

1. In addition to Syncfusion ASP.NET MVC packages, install gulp and glob packages using below commands.

Note: Gulp Installation - `npm install gulp --save`

.NET CLI

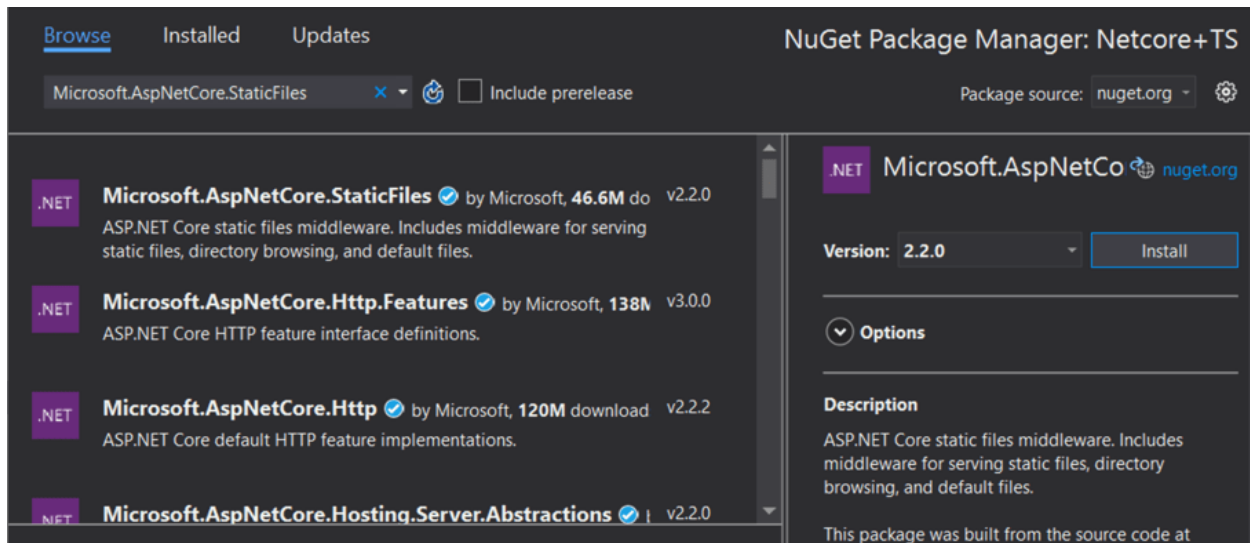
```
npm install gulp@latest --save
```

Note: Glob Installation - `npm install glob --save`

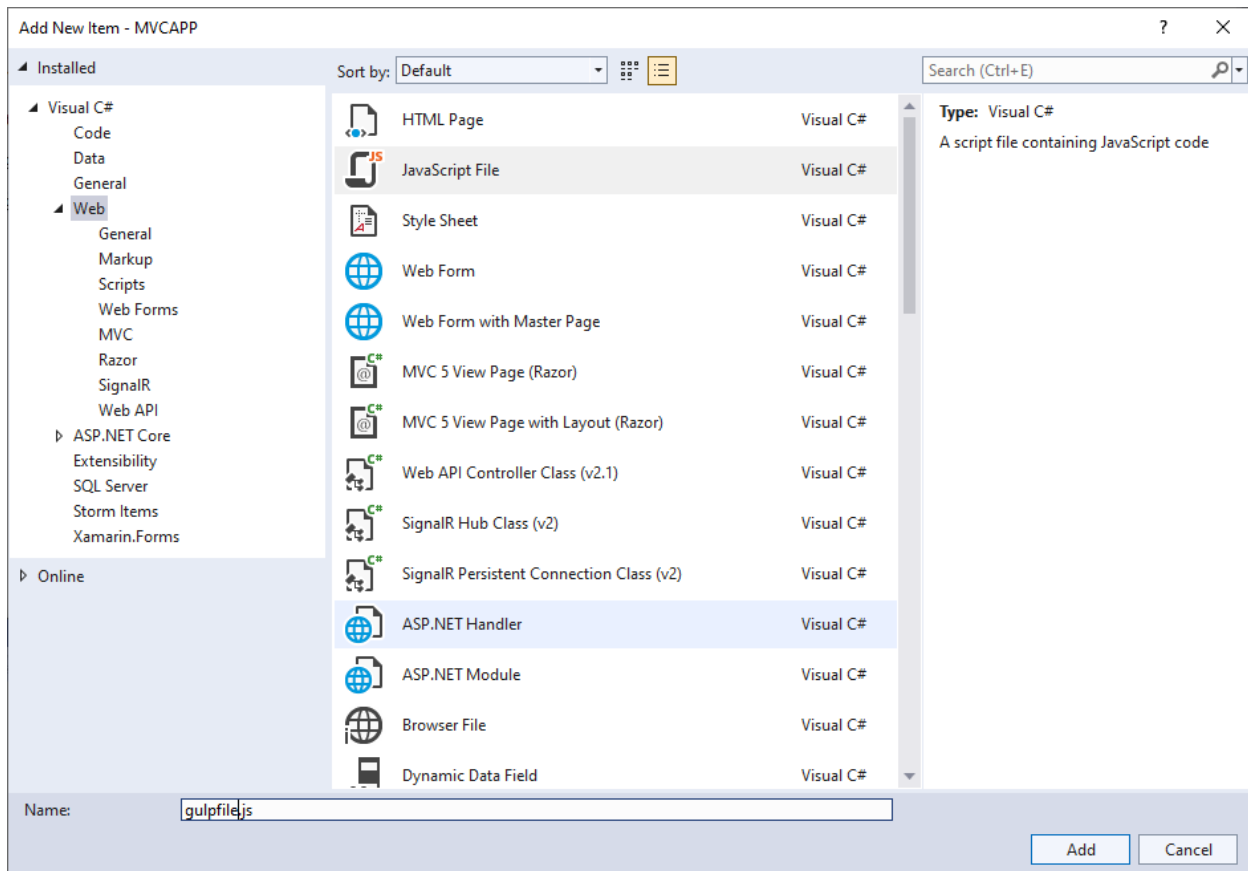
.NET CLI

```
npm install glob@latest --save
```

2.To set up the server, Tools → NuGet Package Manager → Manage NuGet Packages for Solution. Then, search and install "**Microsoft.AspNetCore.StaticFiles**" and "**Microsoft.TypeScript.MSBuild**" packages.



3.Open the Source Explorer and right click on the application name. Then, select “**Add New Item**” menu item to open the “**Add New Item**” window. 4.Select “**ASP.NET Core**” on the left side Tree View and select “**JavaScript File**” in the “**Add New Item**” window. It will include a js file in the root folder of the ASP.NET MVC web application. Rename the js file as “**gulpfile.js**”.



5. Copy the following code snippet and paste it in **gulpfile.js** for automatically copying the script and styles from “**node_modules**” to “**Content**” while building the web application.

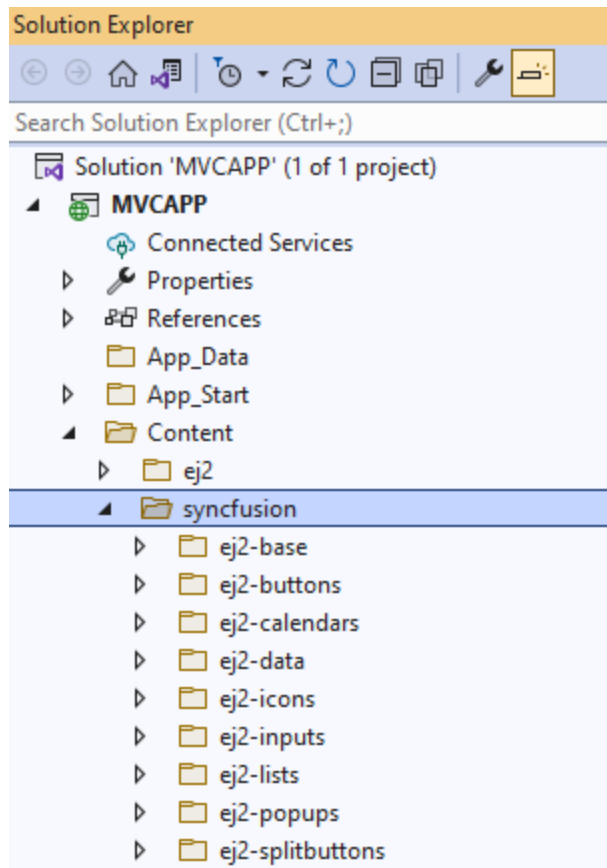
GULPFILE.JS

```

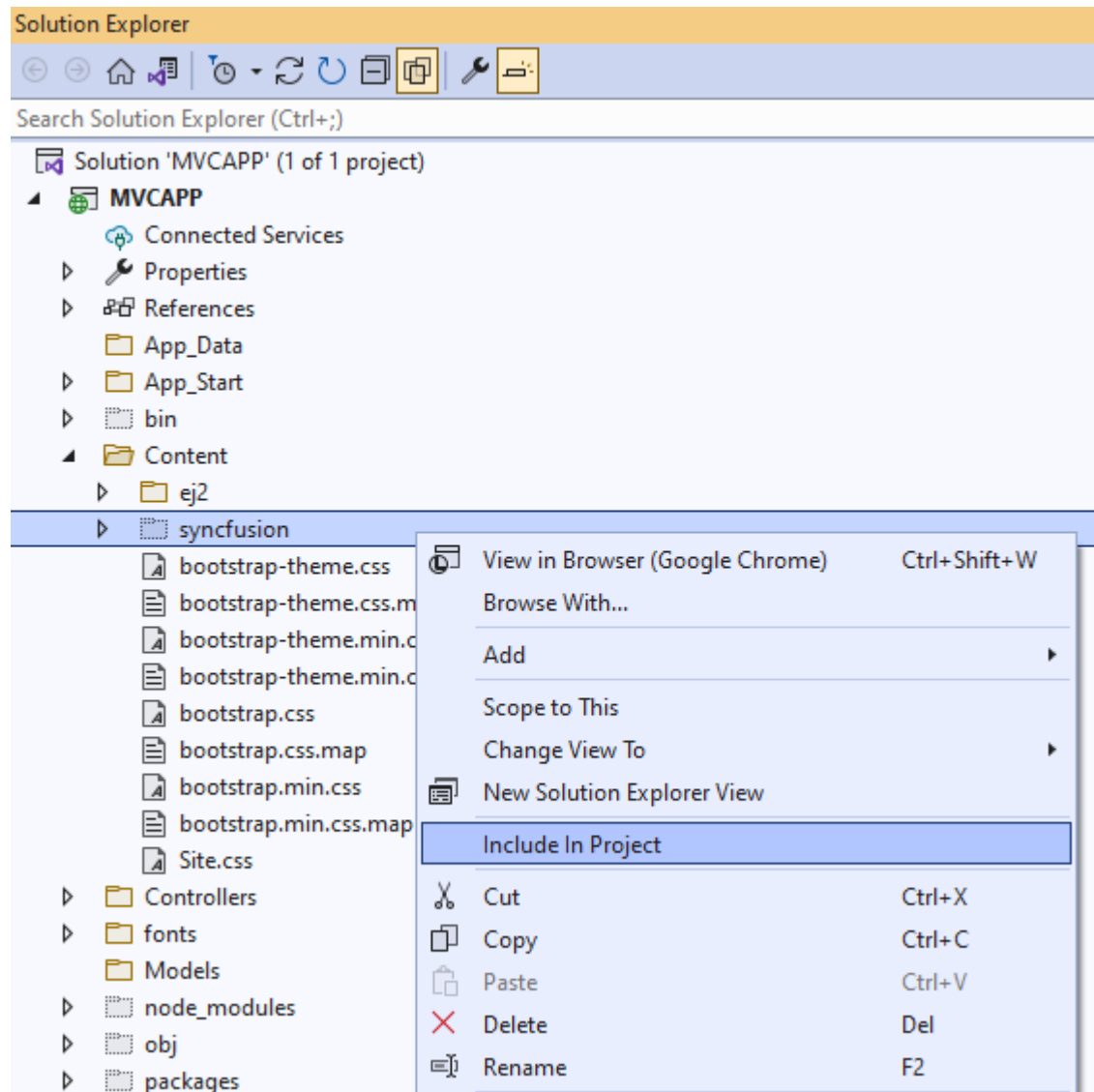
/// <binding BeforeBuild='copy-client-resource'/>
// nodejs requiring statement for importing and using the package in this js
file
var gulp = require('gulp');
var glob = require('glob');
// gulp task for copying file form "node_modules" to "Content" directory
gulp.task("copy-client-resource", function (done) {
  let packagePath = './node_modules/@syncfusion/';
  let destCommonPath = 'Content/syncfusion';
  let installedPackages = glob.sync(`${packagePath}*`);
  for (let insPackage of installedPackages) {
    let packagename = insPackage.replace(packagePath, '');
    gulp.src(`${insPackage}/dist/global/**/*`)
      .pipe(gulp.dest(`${destCommonPath}/${packagename}/`));
    gulp.src(`${insPackage}/styles/**/*.*css`)
      .pipe(gulp.dest(`${destCommonPath}/${packagename}/styles/`));
  }
  done();
});

```

6. Build the ASP.NET MVC web application and notice that a new folder named “**Syncfusion**” is created in “**Content**” folder.



7.Right click the “**Syncfusion**” folder and include in project like below.



8. Add the client-side resource in the element of the `~/Views/Shared/_Layout.cshtml`. Here, scripts and styles of Syncfusion ASP.NET MVC Calendar has been loaded for example.

~/ LAYOUT.CSHTML

```
<head>
<!-- Syncfusion ASP.NET MVC controls styles -->
@Styles.Render("~/Content/syncfusion/ej2-
calendars/styles/calendar/material.css")
@Styles.Render("~/Content/syncfusion/ej2-base/styles/material.css")
@Styles.Render("~/Content/syncfusion/ej2-
buttons/styles/button/material.css")
<!-- Syncfusion ASP.NET MVC controls scripts -->
@Scripts.Render("~/Content/syncfusion/ej2-base/ej2-base.min.js")
@Scripts.Render("~/Content/syncfusion/ej2-calendars/ej2-calendars.min.js")
</head>
```

Custom Resource Generator

The Syncfusion ASP.NET MVC provides an option to generate a control's scripts using the [Custom Resource Generator](#) (CRG) tool for the ASP.NET MVC Controls. To generate the control-wise scripts externally using CRG, refer [here](#).

Custom Resource Generator for Syncfusion ASP.NET MVC

Syncfusion provides an option to generate a selective controls script (JavaScript ES5) and styles by using the [Custom Resource Generator](#) (CRG) web tool for ASP.NET MVC controls from v18.1.0.42.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

Select Themes ▼

Search Components
 ☐ Select all

> <input type="checkbox"/> AccumulationChart	> <input type="checkbox"/> RichTextEditor	<input type="checkbox"/> DatePicker	<input type="checkbox"/> Sidebar
> <input type="checkbox"/> Chart	> <input type="checkbox"/> Schedule	<input type="checkbox"/> DateRangePicker	<input type="checkbox"/> Slider
> <input type="checkbox"/> CircularGauge	> <input type="checkbox"/> Smithchart	<input type="checkbox"/> DateTimePicker	<input type="checkbox"/> SplitButton
> <input type="checkbox"/> Diagram	> <input type="checkbox"/> Sparkline	<input type="checkbox"/> Dialog	<input type="checkbox"/> Switch
> <input type="checkbox"/> DocumentEditor	> <input type="checkbox"/> TreeMap	<input type="checkbox"/> DropDownButton	<input type="checkbox"/> SymbolPalette
> <input type="checkbox"/> Grid	<input type="checkbox"/> Accordion	<input type="checkbox"/> DropDownList	<input type="checkbox"/> Tab
> <input type="checkbox"/> HeatMap	<input type="checkbox"/> AutoComplete	<input type="checkbox"/> PivotFieldList	<input type="checkbox"/> TimePicker
> <input type="checkbox"/> LinearGauge	<input type="checkbox"/> Button	<input type="checkbox"/> MaskedTextBox	<input type="checkbox"/> Toast
> <input type="checkbox"/> ListView	<input type="checkbox"/> Calendar	<input type="checkbox"/> NumericTextBox	<input type="checkbox"/> Toolbar

IMPORT SETTINGS

DOWNLOAD

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

Search and select the control list

Search and select the required Syncfusion controls from the CRG to generate specific set of control resources.

The Syncfusion ASP.NET MVC UI controls can be categorized based on the below characteristics.

- Injectable module supported controls
- Injectable module non-supported controls

Injectable module supported controls are rendered as treeview with checkbox format and the Injectable module non-supported controls are rendered as normal checkbox format in the CRG application.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

Select Themes ▼

Search Components
 ☐ Select all

Injectable module supported components	Injectable module not supported components
<input type="checkbox"/> Accumulation <input type="checkbox"/> RichTextEditor	<input type="checkbox"/> DatePicker <input type="checkbox"/> Sidebar
<input type="checkbox"/> Chart <input type="checkbox"/> Schedule	<input type="checkbox"/> DateRangePicker <input type="checkbox"/> Slider
<input type="checkbox"/> CircularGauge <input type="checkbox"/> Smithchart	<input type="checkbox"/> DateTimePicker <input type="checkbox"/> SplitButton
<input type="checkbox"/> Diagram <input type="checkbox"/> Sparkline	<input type="checkbox"/> Dialog <input type="checkbox"/> Switch
<input type="checkbox"/> DocumentEditor <input type="checkbox"/> TreeMap	<input type="checkbox"/> DropDownButton <input type="checkbox"/> SymbolPalette
<input type="checkbox"/> Grid <input type="checkbox"/> Accordion	<input type="checkbox"/> DropDownList <input type="checkbox"/> Tab
<input type="checkbox"/> HeatMap <input type="checkbox"/> AutoComplete	<input type="checkbox"/> PivotFieldList <input type="checkbox"/> TimePicker
<input type="checkbox"/> LinearGauge <input type="checkbox"/> Button	<input type="checkbox"/> MaskedTextBox <input type="checkbox"/> Toast
<input type="checkbox"/> ListView <input type="checkbox"/> Calendar	<input type="checkbox"/> NumericTextBox <input type="checkbox"/> Toolbar

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

Refer to the following steps to search and select the controls in CRG.

1. Open [Syncfusion Custom Resource Generator](#) (CRG) application. 2. Type the required control name in the search bar and select the check box. The dependency of the selected control is resolved in the application itself, so it does not need to choose each dependent control by manually.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

Select Themes ▼

🔍 button



You have selected 1 Component



Button

☐ DropDownButton

☐ SplitButton

☐ RadioButton

📄 IMPORT SETTINGS

📄 DOWNLOAD

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

3. Click the expand icon and select the required feature for the injectable module supported controls.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

Select Themes

You have selected 2 Components

→

Grid

☒

Filter

☐

Page

☐

Selection

☐

Sort

☐

Group

☐

Reorder

☐

RowDD

☐

DetailRow

☐

Toolbar

☐

Aggregate

☐

Search

☐

VirtualScroll

☐

☐

Accordion

☐

AutoComplete

☒

Button

☐

Calendar

☐

CheckBox

☐

ColorPicker

☐

ComboBox

☐

ContextMenu

☐

DropDownList

☐

PivotFieldList

☐

MaskedTextBox

☐

NumericTextBox

☐

Overview

☐

Pager

☐

RadioButton

☐

RecurrenceEditor

☐

Tab

☐

TimePicker

☐

Toast

☐

Toolbar

☐

Tooltip

☐

TreeView

☐

Uploader

[IMPORT SETTINGS](#) [DOWNLOAD](#)

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

4. If the entire modules of the controls are needed, then click the specific control's check box to select all injectable modules.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

Select Themes ▼

You have selected 2 Components

<input checked="" type="checkbox"/> Grid	<input type="checkbox"/> Accordion	<input type="checkbox"/> DropDownList	<input type="checkbox"/> Tab
<input checked="" type="checkbox"/> Filter	<input type="checkbox"/> AutoComplete	<input type="checkbox"/> PivotFieldList	<input type="checkbox"/> TimePicker
<input checked="" type="checkbox"/> Page	<input type="checkbox"/> Button	<input type="checkbox"/> MaskedTextBox	<input type="checkbox"/> Toast
<input checked="" type="checkbox"/> Selection	<input type="checkbox"/> Calendar	<input type="checkbox"/> NumericTextBox	<input type="checkbox"/> Toolbar
<input checked="" type="checkbox"/> Sort	<input type="checkbox"/> CheckBox	<input type="checkbox"/> Overview	<input type="checkbox"/> Tooltip
<input checked="" type="checkbox"/> Group	<input type="checkbox"/> ColorPicker	<input type="checkbox"/> Pager	<input type="checkbox"/> TreeView
<input checked="" type="checkbox"/> Reorder	<input type="checkbox"/> ComboBox	<input type="checkbox"/> RadioButton	<input type="checkbox"/> Uploader
<input checked="" type="checkbox"/> RowDD	<input type="checkbox"/> ContextMenu	<input type="checkbox"/> RecurrenceEditor	
<input checked="" type="checkbox"/> DetailRow			
<input checked="" type="checkbox"/> Toolbar			
<input checked="" type="checkbox"/> Aggregate			
<input checked="" type="checkbox"/> Search			
<input checked="" type="checkbox"/> VirtualScroll			
<input checked="" type="checkbox"/> Edit			

[IMPORT SETTINGS](#) [DOWNLOAD](#)

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

5. Select the required built-in themes from the **Select Themes** option. This provides an option to select more than one theme.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

You have selected 2 Components

<input checked="" type="checkbox"/> Grid	<input type="checkbox"/> Accordion	<input type="checkbox"/> DropDownList
<input checked="" type="checkbox"/> Filter	<input type="checkbox"/> AutoComplete	<input type="checkbox"/> PivotFieldList
<input checked="" type="checkbox"/> Page	<input checked="" type="checkbox"/> Button	<input type="checkbox"/> MaskedTextBox
<input checked="" type="checkbox"/> Selection	<input type="checkbox"/> Calendar	<input type="checkbox"/> NumericTextBox
<input checked="" type="checkbox"/> Sort	<input type="checkbox"/> CheckBox	<input type="checkbox"/> Overview
<input checked="" type="checkbox"/> Group	<input type="checkbox"/> ColorPicker	<input type="checkbox"/> Pager
<input checked="" type="checkbox"/> Reorder	<input type="checkbox"/> ComboBox	<input type="checkbox"/> RadioButton
<input checked="" type="checkbox"/> RowDD	<input type="checkbox"/> ContextMenu	<input type="checkbox"/> RecurrenceEditor
<input checked="" type="checkbox"/> DetailRow		
<input checked="" type="checkbox"/> Toolbar		
<input checked="" type="checkbox"/> Aggregate		
<input checked="" type="checkbox"/> Search		
<input checked="" type="checkbox"/> VirtualScroll		
<input checked="" type="checkbox"/> Edit		

Material, Fabric

Select Themes

☐ Select All

☒ Material

☒ Fabric

☐ Bootstrap

IMPORT SETTINGS

DOWNLOAD

[Contact Us](#) | [About Us](#)

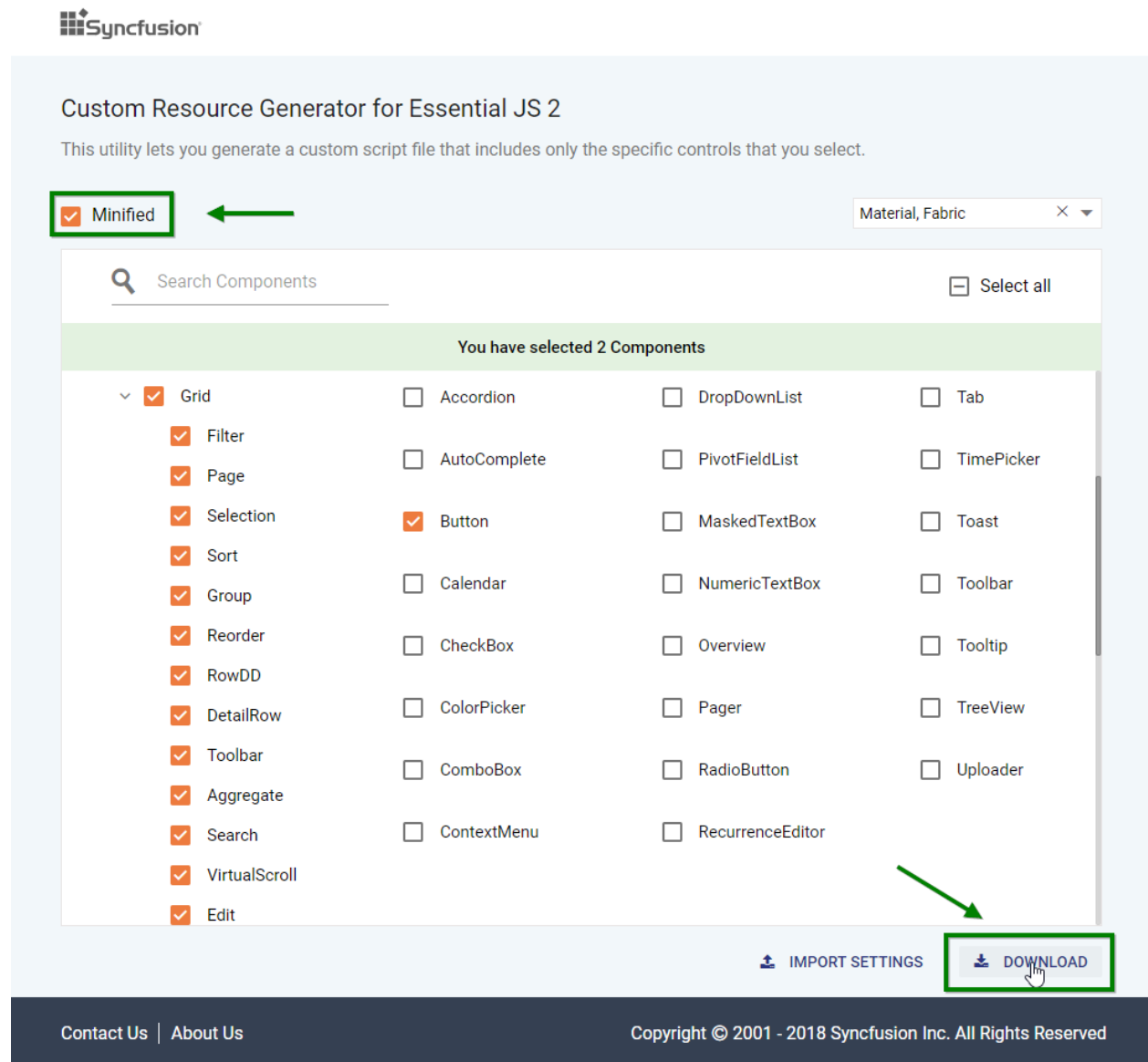
Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

Download the selected control resources

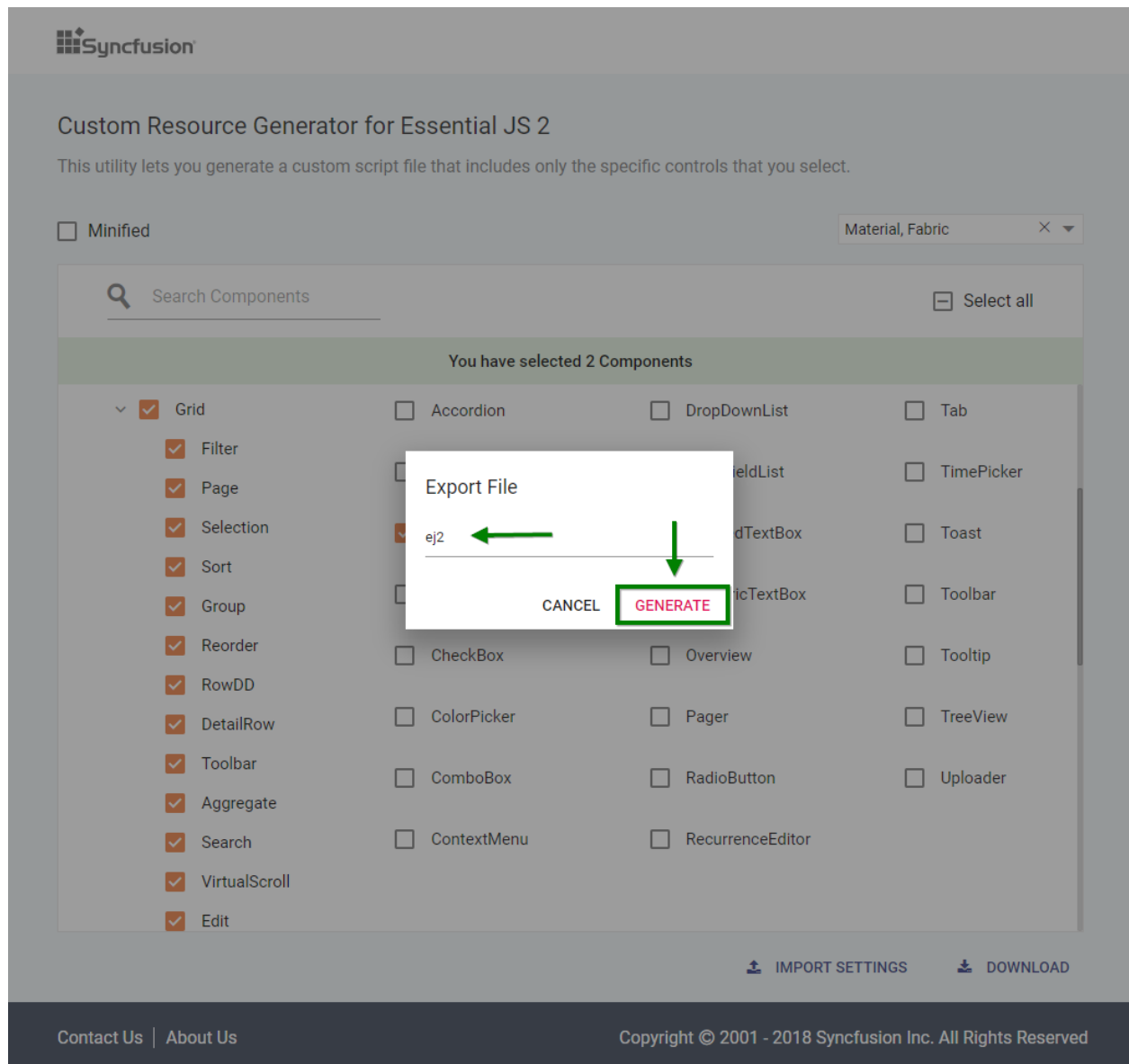
After selecting the required control resources, download the custom script and styles from CRG.

Refer to the following steps to download the custom resources in CRG.

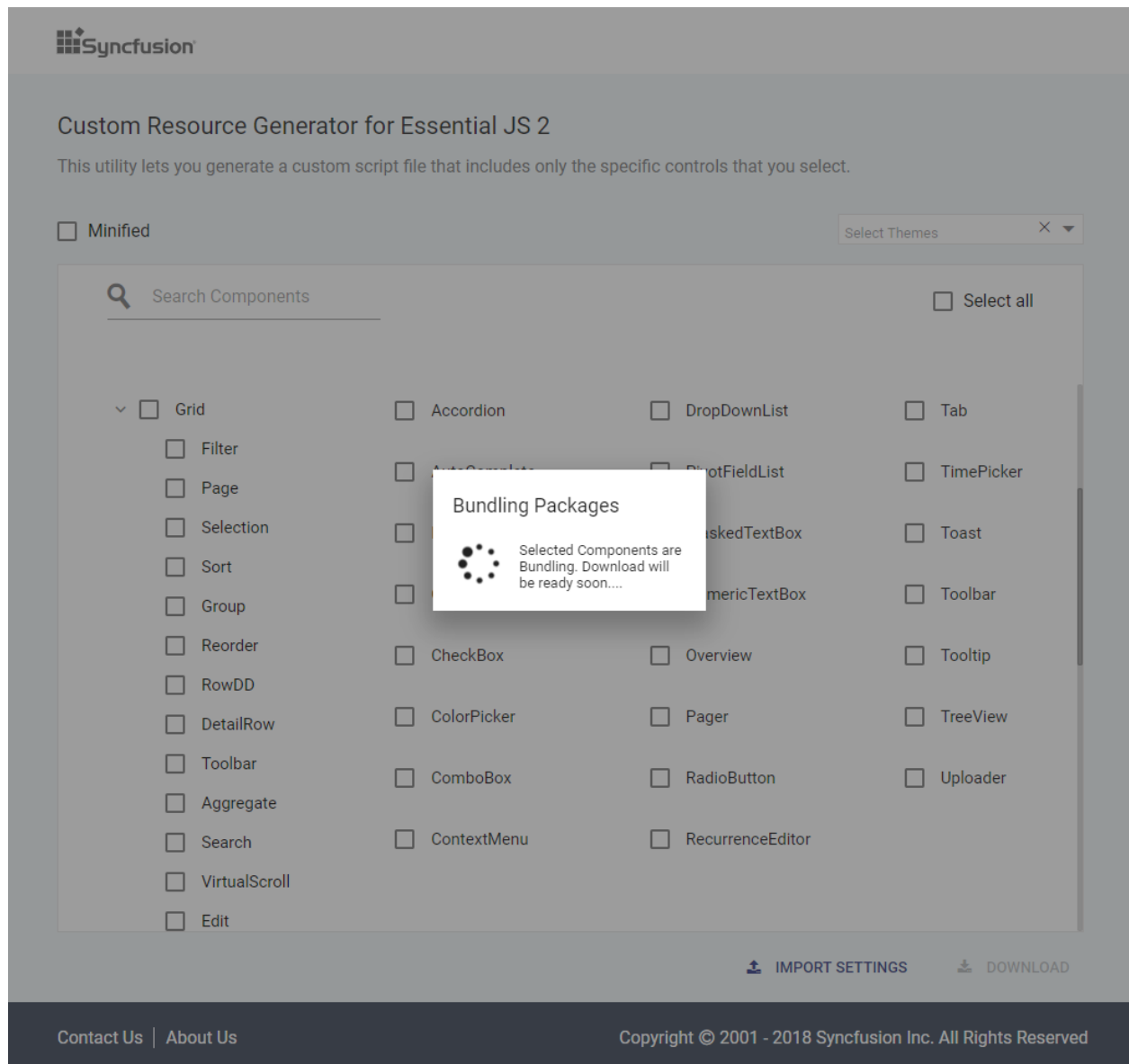
1. Click the **DOWNLOAD** button at the bottom of the page. Select the **Minified** option to generate the minified file output for production.



2. Change the file name as you desire and click **GENERATE** button in the pop-up.

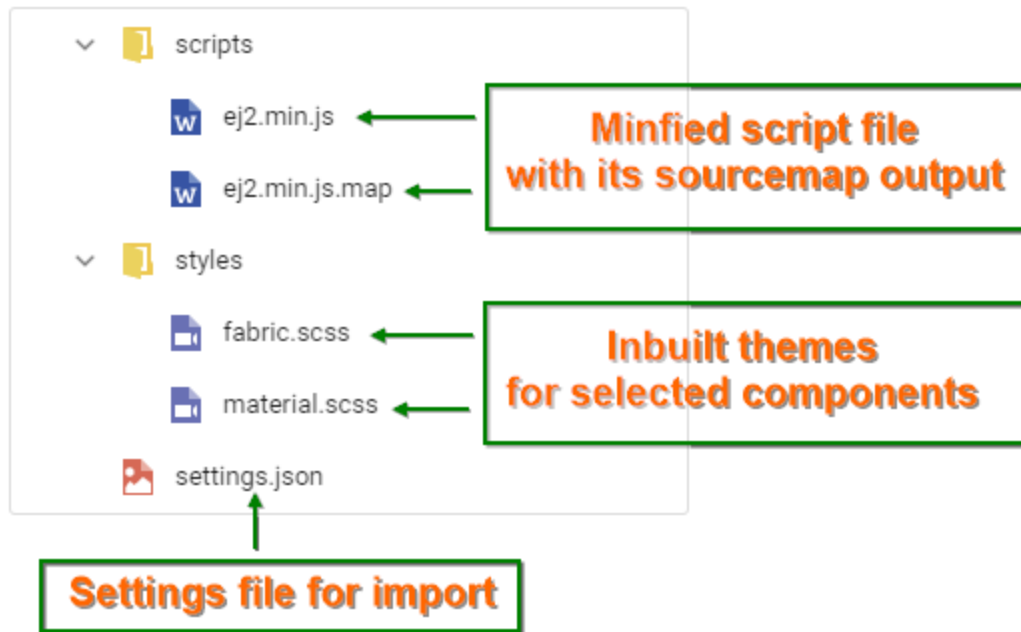


- Now, the bundling process for the selected controls will be started, and the output will be downloaded as a zip file.



4. The final output contains the script and styles for the selected controls and a **import.json** file, which stores the current settings.

Note: You can use theme files in the **customized** folder if you don't want google fonts references. For now, Material and tailwind themes only generate like below.



How to use custom resources in the ASP.NET MVC application

1. Extract the downloaded script into `~/Content` folder of the ASP.NET MVC web application.
2. Refer the script and style in `~/Views/Shared/_Layout.cshtml` file.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>@ViewBag.Title - My ASP.NET Application</title>
7      @Styles.Render("~/Content/css")
8      @Scripts.Render("~/bundles/modernizr")
9      <!-- Syncfusion ASP.NET Core controls styles -->
10     @Styles.Render("~/Content/styles/material.css")
11     <!-- Syncfusion ASP.NET Core controls scripts -->
12     @Scripts.Render("~/Content/scripts/ej2.min.js")
13

```

3. Run the application and it will load the resources with application required controls.

Import previously generated settings into the CRG

To add more controls or upgrade the latest Syncfusion ASP.NET MVC library resources, it is not necessary to generate it from the scratch in the CRG. Just import the old **import.json** file and make the changes, then download it again from the CRG application.

Refer to the following steps to import previous **import.json** file in the CRG.

1. Click the **IMPORT SETTINGS** button at the bottom of the page.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☐ Minified

Select Themes ▼

☐ Select all

> <input type="checkbox"/> AccumulationChart	> <input type="checkbox"/> RichTextEditor	<input type="checkbox"/> DatePicker	<input type="checkbox"/> Sidebar
> <input type="checkbox"/> Chart	> <input type="checkbox"/> Schedule	<input type="checkbox"/> DateRangePicker	<input type="checkbox"/> Slider
> <input type="checkbox"/> CircularGauge	> <input type="checkbox"/> Smithchart	<input type="checkbox"/> DateTimePicker	<input type="checkbox"/> SplitButton
> <input type="checkbox"/> Diagram	> <input type="checkbox"/> Sparkline	<input type="checkbox"/> Dialog	<input type="checkbox"/> Switch
> <input type="checkbox"/> DocumentEditor	> <input type="checkbox"/> TreeMap	<input type="checkbox"/> DropDownButton	<input type="checkbox"/> SymbolPalette
> <input type="checkbox"/> Grid	<input type="checkbox"/> Accordion	<input type="checkbox"/> DropDownList	<input type="checkbox"/> Tab
> <input type="checkbox"/> HeatMap	<input type="checkbox"/> AutoComplete	<input type="checkbox"/> PivotFieldList	<input type="checkbox"/> TimePicker
> <input type="checkbox"/> LinearGauge	<input type="checkbox"/> Button	<input type="checkbox"/> MaskedTextBox	<input type="checkbox"/> Toast
> <input type="checkbox"/> ListView	<input type="checkbox"/> Calendar	<input type="checkbox"/> NumericTextBox	<input type="checkbox"/> Toolbar

 [IMPORT SETTINGS](#)

[DOWNLOAD](#)

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

2. Upload the **import.json** file so that the previously stored data will be restored in the CRG application. Now, add more controls and export the resources again.



Custom Resource Generator for Essential JS 2

This utility lets you generate a custom script file that includes only the specific controls that you select.

☒ Minified

Material, Fabric

You have selected 2 Components

> <input type="checkbox"/> AccumulationCl	> <input type="checkbox"/> RichTextEditor	<input type="checkbox"/> DatePicker	<input type="checkbox"/> Sidebar
> <input type="checkbox"/> Chart	> <input type="checkbox"/> Schedule	<input type="checkbox"/> DateRangePicker	<input type="checkbox"/> Slider
> <input type="checkbox"/> CircularGauge	> <input type="checkbox"/> Smithchart	<input type="checkbox"/> DateTimePicker	<input type="checkbox"/> SplitButton
> <input type="checkbox"/> Diagram	> <input type="checkbox"/> Sparkline	<input type="checkbox"/> Dialog	<input type="checkbox"/> Switch
> <input type="checkbox"/> DocumentEditor	> <input type="checkbox"/> TreeMap	<input type="checkbox"/> DropDownButton	<input type="checkbox"/> SymbolPalette
v <input checked="" type="checkbox"/> Grid <div> <input checked="" type="checkbox"/> Filter <input checked="" type="checkbox"/> Page <input checked="" type="checkbox"/> Selection <input checked="" type="checkbox"/> Sort <input checked="" type="checkbox"/> Group </div>	<input type="checkbox"/> Accordion <input type="checkbox"/> AutoComplete <input checked="" type="checkbox"/> Button <input type="checkbox"/> Calendar	<input type="checkbox"/> DropDownList <input type="checkbox"/> PivotFieldList <input type="checkbox"/> MaskedTextBox <input type="checkbox"/> NumericTextBox	<input type="checkbox"/> Tab <input type="checkbox"/> TimePicker <input type="checkbox"/> Toast <input type="checkbox"/> Toolbar

[Contact Us](#) | [About Us](#)

Copyright © 2001 - 2018 Syncfusion Inc. All Rights Reserved

Strongly-Typed HTML Helper

The Syncfusion editor controls supports the strongly typed HTML helpers represented by lambda expressions that have the model or template passed into the view. The Extension method is used to get a value from the model.

The Strongly-Typed HTML helper (i.e., NumericTextBox) takes lambda as a parameter that tells the helper, which element of the model to be used in the typed view.

The Strongly typed views are used for rendering specific types of model objects, instead of using the general ViewData structure.

The following list of controls supports the Strongly-Typed HTML Helper

- Autocomplete
- Checkbox and Radio Button
- DatePicker

- DateTimePicker
- DropDownList
- Numeric, Currency and Percentage Textbox
- RichText Editor
- TimePicker

The following steps explain how to use the strongly typed helpers to create a Numeric Textbox.

- The NumericTextBox control supports strongly typed HTML helpers that uses lambda expression to refer to the models or view models passed to a view template. These helpers allows you to define the value of the NumericTextBoxFor from the model.

Add a class name “EditorValue” in the Models folder and replace the code with the following code:

```
`csharp
public class EditorValue
{
    public int number { set; get; }
    public EditorValue(int value)
    {
        number = value;
    }
    public EditorValue() { }
}
`
```

- Create an action method that renders the NumericTextBox on the view page, and passes the model to be bound to the view page.

```
`csharp
using Syncfusion.EJ2.Inputs;

public ActionResult Index()
{
    //Editor For
    NumericTextBox editor = new NumericTextBox();
    editor.Enabled = false;
    ViewData["editmodel"] = editor;
    return View(new EditorValue(66));
}
```

- In View, invoke the strongly typed `NumericTextBoxFor` helper with the lambda expression to set the default value.

```
`html
<div>
@using (Html.BeginForm())
{
@Html.EJS().NumericTextBoxFor(model => model.number,
(Syncfusion.EJ2.Inputs.NumericTextBox)ViewData["editmodel"]).Render();
@Html.EJS().Button("btn").Content("Post").Render()
}
</div>
```

10.00



POST

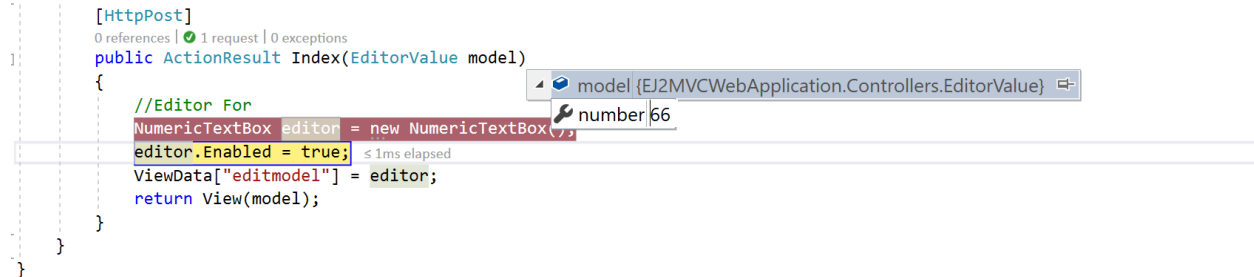
The following steps explain how to get the values by using the Scaffolding methods in Post back.

- Create an action method, `FormPost` that handles the Post request and processes the data. In the action method, you can pass the model as the parameter and that model has the `NumericTextBox`'s value.

```
`csharp
[HttpPost]
public ActionResult Index(EditorValue model)
{
//Editor For
NumericTextBox editor = new NumericTextBox();
editor.Enabled = true;
ViewData["editmodel"] = editor;
return View(model);
}
```

```
}
,
```

On clicking the button, the Post method will be triggered. In that, the selected value will be obtained as follows.



```
[HttpPost]
0 references | 1 request | 0 exceptions
public ActionResult Index(EditorValue model)
{
    //Editor For
    NumericTextBox editor = new NumericTextBox(, ,
    editor.Enabled = true; ≤ 1ms elapsed
    ViewData["editmodel"] = editor;
    return View(model);
}
```

Server Side Validation

In the server-side validation, the page must be submitted via a postback to validate on the server and if the model data is not valid, the server sends a response back to the client.

The best way to validate a model is by using the Data Annotations that has a set of attributes and classes defined in the System.ComponentModel.DataAnnotations assembly.

Step 1: Add the following namespace to the “EditorValue” model.

```
`csharp
using System.ComponentModel.DataAnnotations;
,
```

The Data Annotations allows to decorate model classes with the metadata. This metadata describes a set of rules that are used to validate a property.

The following Data Annotation attributes are used for the Numeric Textbox.

Required: Indicates that the property is a required field.

Step 2: Next, Update the number property of the “EditorValue” class as “Required Field” by adding the following line.

```
`csharp
public class EditorValue
{
    [Required(ErrorMessage = "Numeric Textbox is Required")]
    public int number { set; get; }
}
,
```

Step 3: Modify the view page as follows:

```
`html
<div>
```

```
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)
    @Html.EJS().NumericTextBoxFor(model => model.number,
    (Syncfusion.EJ2.Inputs.NumericTextBox)ViewData["editmodel"]).Render()
    @Html.ValidationMessageFor(model => model.number)
    @Html.EJS().Button("btn").Content("Post").Render()
}
</div>
```

When you press the “POST” button on this page then it will post the data to the server and the code written with in EditorFor action will validate the NumericTextBox value by checking the ModelState.IsValid property. If the NumericTextBox value is not selected, the ModelState.IsValid will return false and display error message.

0.00



Numeric Textbox is Required

POST

[Accessibility support with ADA compliance in ASP.NET MVC controls](#)

All the Syncfusion ASP.NET MVC controls follow the WAI-ARIA accessibility standard by default. This enables to build accessible web applications, which are fully navigable by users with disabilities.

[Section 508](#)

Section 508 is part of the U.S. Rehabilitation Act of 1973 introduced in 1998, requires Federal agencies to make their electronic and information technology (EIT) accessible to people with disabilities. This law defines the need for both members of the public and Federal employees in all Federal agencies to have access to, develop, maintain, procure, or use electronic and information technology. Under Section 508, Federal agencies must develop a website that can be used by people with disabilities.

Note: All our Syncfusion ASP.NET MVC controls comply with the standard of Section 508 accessibility by default. This enables you to build accessible web applications, which are fully navigable by users with disabilities.

[Keyboard navigation](#)

Keyboard navigation support enables users to interact with controls using keyboard shortcuts. Each control has its own set of shortcuts. Refer to the following documents to get the details of each control.

- [Accordion](#)
- [AutoComplete](#)
- [ButtonGroup](#)
- [Calendar](#)
- [CheckBox](#)
- [Chips](#)
- [ColorPicker](#)
- [ComboBox](#)
- [ContextMenu](#)
- [DatePicker](#)
- [DateRangePicker](#)
- [DateTimePicker](#)
- [Dialog](#)
- [Document Editor](#)
- [DropDownButton](#)
- [DropDownList](#)
- [DropDownTree](#)
- [FileManager](#)
- [Gantt](#)
- [Grid](#)
- [List Box](#)
- [List View](#)
- [Menu](#)
- [MultiSelect](#)
- [NumericTextBox](#)
- [PDFViewer](#)
- [Pivot Table](#)
- [ProgressButton](#)
- [Range Slider](#)
- [RichTextEditor](#)
- [Schedule](#)
- [SplitButton](#)
- [Splitter](#)
- [SpreadSheet](#)
- [Switch](#)
- [Tabs](#)
- [TimePicker](#)
- [ToolBar](#)
- [Tree Grid](#)
- [TreeView](#)
- [Uploader](#)

Localization (Multi-Language) support in ASP.NET MVC

Localization (L10N) is the process of adapting application controls and content to the desired language with its corresponding region. This page shows, how to use the Localization feature in the ASP.NET MVC application.

Localization of Syncfusion ASP.NET MVC Controls

Syncfusion controls can be localized using the culture based JSON files. You can find the default and culture based localization files in the following GitHub repository.

Note: You can get default and culture based JSON files from [GitHub](#).

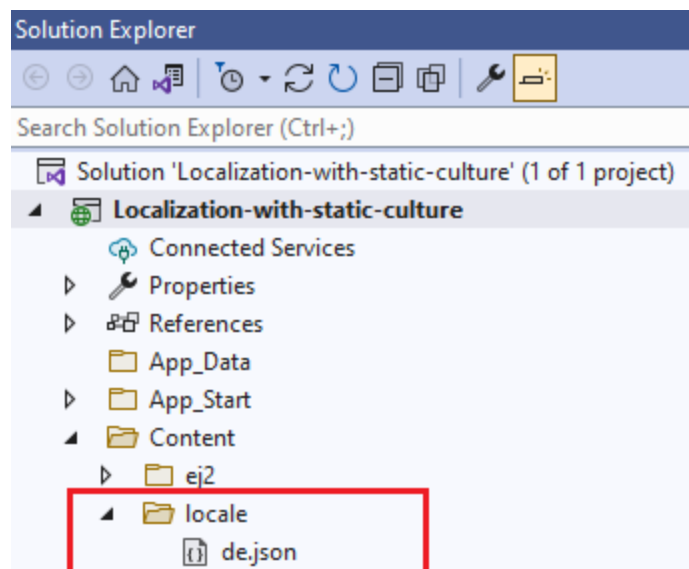
Adding culture based JSON files

Syncfusion [locale data](#) is also available as npm package. So, you can install it through the below command in the application root directory. Once the package is installed, you can find the culture specific JSON data under the location `node_modules/@syncfusion/ej2-locale/src`.

CMD

```
npm i @syncfusion/ej2-locale
```

Once the `ej2-locale` package has installed, create a folder `locale` inside the `Content` folder. Then copy the specific culture files from `node_modules/@syncfusion/ej2-locale/src` location and paste it inside the `~/Content/locale` folder like below. The culture file contains all Syncfusion ASP.NET MVC controls locale text.



Statically set the culture

If you don't want to change culture dynamically, you can specify the static culture using `load` function of `L10n` class in `~/Views/Shared/_Layout.cshtml` file as in the below code.

~/_LAYOUT.CSHTML

```
<body>
...
<script>
var ajax = new ej.base.Ajax(location.origin + '/Content/locale/de.json',
'GET', false); //load the de json culture file
ajax.send().then((e) => {
var loader = JSON.parse(e);
ej.base.L10n.load(
loader
);
});
```

```

ej.base.setCulture('de');           //Set the culture for the ASP.NET MVC
controls
});
</script>
</body>

```

Now, you can add the Syncfusion ASP.NET MVC control in `~/Views/Home/Index.cshtml` page.

CSHTML

```

@Html.EJS().Grid("RemoteData").DataSource(dataManger =>
{
    dataManger.Url("https://ej2services.syncfusion.com/production/web-
services/api/Orders").CrossDomain(true).Adaptor("ODataV4Adaptor");
}).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("160").Add();
    col.Field("EmployeeID").HeaderText("Employee
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("Freight").HeaderText("Freight").Width("150").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().AllowGrouping().Render()

```

Ziehen Sie eine Spaltenüberschrift hierher, um die Spalte zu gruppieren				
Order ID	Customer ID	Employee ID	Freight	Ship Country
10001	ALFKI	1	\$2.30	Denmark
10002	ANATR	3	\$3.30	Brazil
10003	ANTON	2	\$4.30	Germany
10004	BLONP	4	\$5.30	Austria
10005	BOLID	5	\$6.30	Switzerland
10006	ALFKI	2	\$4.60	Denmark
10007	ANATR	4	\$6.60	Brazil
10008	ANTON	3	\$8.60	Germany
10009	BLONP	5	\$10.60	Austria
10010	BOLID	6	\$12.60	Switzerland
10011	ALFKI	3	\$6.90	Denmark
10012	ANATR	5	\$9.90	Brazil

<< < 1 2 3 4 > >>
 1 von 4 Seiten (45 Datensätze)

Note: [View sample in GitHub](#)

Dynamically set the culture

The culture can be set dynamically based on user's preference. The following example demonstrates how to dynamically change the culture in the ASP.NET MVC application.

Implement a culture change dynamically using the dropdown in the application like below.

CSHTML

```

<div class="dropdown">
@Html.EJS().DropDownList("culture-
switch").DataSource((IEnumerable<Object>)ViewBag.data).Change("onCultureChan
ge").Width("20%").Index(0).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "Text", Value =
"ID" }).Render()
</div>
<script>
function onCultureChange(e) {
var culture = e.value;
var ajax = new ej.base.Ajax(location.origin + '/Content/locale/' + culture +
'.json', 'GET', false); //load the json culture file
ajax.send().then((e) => {
var loader = JSON.parse(e);
ej.base.L10n.load(
loader
);
ej.base.setCulture(culture); //Set the culture for the ASP.NET Core
controls
});
}
</script>

```

HOMECONTROLLER.CS

```

using System.Web.Mvc;
using Localization_with_dynamic_culture.Models;
namespace Localization_with_dynamic_culture.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.data = new CultureDetails().Cultures();
return View();
}
...
}
}

```

Create a **CultureDetails** model page and add the following code for dropdown data in the `~/Models/CultureDetails.cs` page.

~/CULTUREDETAILS.CS

```

using System.Web;
namespace Localization_with_dynamic_culture.Models
{
public class CultureDetails
{
public string ID { get; set; }
public string Text { get; set; }
public List<CultureDetails> Cultures()
{
List<CultureDetails> Culture = new List<CultureDetails>();
Culture.Add(new CultureDetails() { ID = "en-US", Text = "English" });
}
}
}

```

```

Culture.Add(new CultureDetails() { ID = "de", Text = "Germany" });
Culture.Add(new CultureDetails() { ID = "fr", Text = "French" });
Culture.Add(new CultureDetails() { ID = "zh", Text = "Chinese" });
return Culture;
}
}
}

```

Germany ▼

Ziehen Sie eine Spaltenüberschrift hierher, um die Spalte zu gruppieren				
Order ID	Customer ID	Employee ID	Freight	Ship Country
10001	ALFKI	1	\$2.30	Denmark
10002	ANATR	3	\$3.30	Brazil
10003	ANTON	2	\$4.30	Germany
10004	BLONP	4	\$5.30	Austria
10005	BOLID	5	\$6.30	Switzerland
10006	ALFKI	2	\$4.60	Denmark
10007	ANATR	4	\$6.60	Brazil
10008	ANTON	3	\$8.60	Germany
10009	BLONP	5	\$10.60	Austria
10010	BOLID	6	\$12.60	Switzerland
10011	ALFKI	3	\$6.90	Denmark
10012	ANATR	5	\$9.90	Brazil

<< < 1 2 3 4 > >>

1 von 4 Seiten (45 Datensätze)

Note: [View sample in GitHub](#)

Changing current locale

Current locale can be changed for all the Syncfusion ASP.NET MVC controls in the application by invoking `setCulture` function with the desired culture name and set locale property. You can change the different culture in Syncfusion ASP.NET MVC controls by setting [locale](#) property with culture codes.

The following example demonstrates the Grid in Deutsch culture.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource(dataManger =>
{
    dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/Orders").CrossDomain(true).Adaptor("ODataV4Adaptor");
}).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().AllowGrouping().Locale("de-DE").PageSettings(page =>
page.PageSize(6)).Render()
<script>
ej.base.L10n.load({

```

```

'de-DE': {
  'grid': {
    'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
    'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die Gruppe ihre
    Spalte',
    'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
    'EmptyDataSourceError': 'DataSource darf bei der Erstaustauslastung nicht leer
    sein, da Spalten aus der dataSource im AutoGenerate Spaltenraster',
    'Item': 'Artikel',
    'Items': 'Artikel'
  },
  'pager': {
    'currentPageInfo': '{0} von {1} Seiten',
    'totalItemsInfo': '({0} Beiträge)',
    'firstPageTooltip': 'Zur ersten Seite',
    'lastPageTooltip': 'Zur letzten Seite',
    'nextPageTooltip': 'Zur nächsten Seite',
    'previousPageTooltip': 'Zurück zur letzten Seit',
    'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-Elementen',
    'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-Elementen'
  }
}
});
ej.base.setCulture('de');
</script>

```

Note: Before changing a culture globally, ensure that locale text for the concerned culture is loaded through `L10n.load` function.

See also

- [Globalization](#)

Globalization in ASP.NET MVC Control

Globalization is the combination of adapting the control to various languages by parsing and formatting the date or numbers (Internationalization (L18N)), adding cultural-specific customizations and translating the text (Localization (L10N)). The American English (en-US) locale is set as default *culture* and `USD` is set as default *currencyCode* for all Syncfusion ASP.NET MVC controls.

Loading Culture Data

It requires the following [CLDR](#) data to be loaded using `loadCldr` function for cultures other than `en-US`.

File Name	Path
ca-gregorian	cldr/main/en/ca-gregorian.json
timeZoneNames	cldr/main/en/timeZoneNames.json
numbers	cldr/main/en/numbers.json
numberingSystems	cldr/supplemental/numberingSystems.json
currencies	cldr/main/en/currencies.json

Note: For en, dependency files are already loaded in the library.

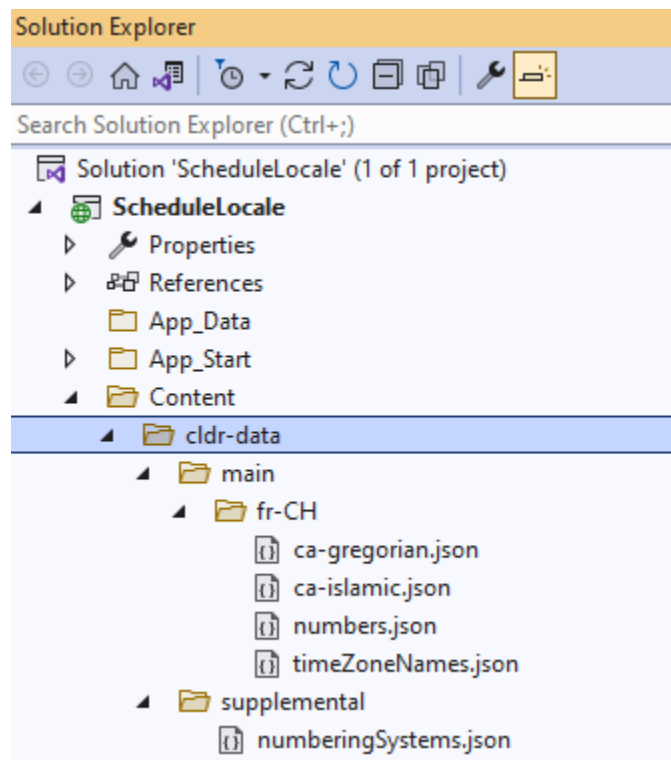
Installing CLDR Data and enable localization in schedule control

1. CLDR data is available as npm package. So, you can install it through the below command in the application root directory. Once the package is installed, you can find the culture specific JSON data under the location `node_modules/cldr-data`.

CMD

```
npm install cldr-data
```

2. Once the CLDR-Data installed create a folder `cldr-data` inside the `Content` folder. Then create the folder directory like shown below in the structure inside the `Content` folder. `Content/cldr-data/supplemental` `Content/cldr-data/main` 3. The files named as below are required to setup the specific culture to the Schedule control. `numberingSystems.json` `ca-gregorian.json` `numbers.json` `timeZoneNames.json` * `ca-islamic.json` 4. The file named `numberingSystems.json` is available in the location `nodemodules/cldr-data/supplemental` which is common for all the cultures. Now, you can move this file to the location `Content/cldr-data/supplemental`. 5. The other required files mentioned above are available in the location `nodemodules/cldr-data/main/culturecode`. In this location every culture has the culture files inside the folder named as its language culture code. For example if the German culture is loaded, then the German culture files could be found inside the location `nodemodules/cldr-data/main/fr-CH`. Now, create a folder named `fr-CH` inside the location `Content/cldr-data/main` and move the files inside it.



6. Now, use the below `loadCultureFiles` method to load the culture specific CLDR JSON data.

CSHTML

```

<script>
loadCultureFiles('fr-CH');
function loadCultureFiles(name) {
var files = ['ca-gregorian.json', 'numberingSystems.json', 'numbers.json',
'timeZoneNames.json', 'ca-islamic.json'];
var loader = ej.base.loadCldr;
var loadCulture = function (prop) {
var val, ajax;
if (files[prop] === 'numberingSystems.json') {
ajax = new ej.base.Ajax(location.origin + '/../Content/cldr-
data/supplemental/' + files[prop], 'GET', false);
} else {
ajax = new ej.base.Ajax(location.origin + '/../Content/cldr-data/main/' +
name + '/' + files[prop], 'GET', false);
}
ajax.onSuccess = function (value) {
val = value;
};
ajax.send();
loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
loadCulture(prop);
}
}
</script>

```

7.The following code example lets you to set the culture to the Schedule control by using the locale property.

CSHTML

```

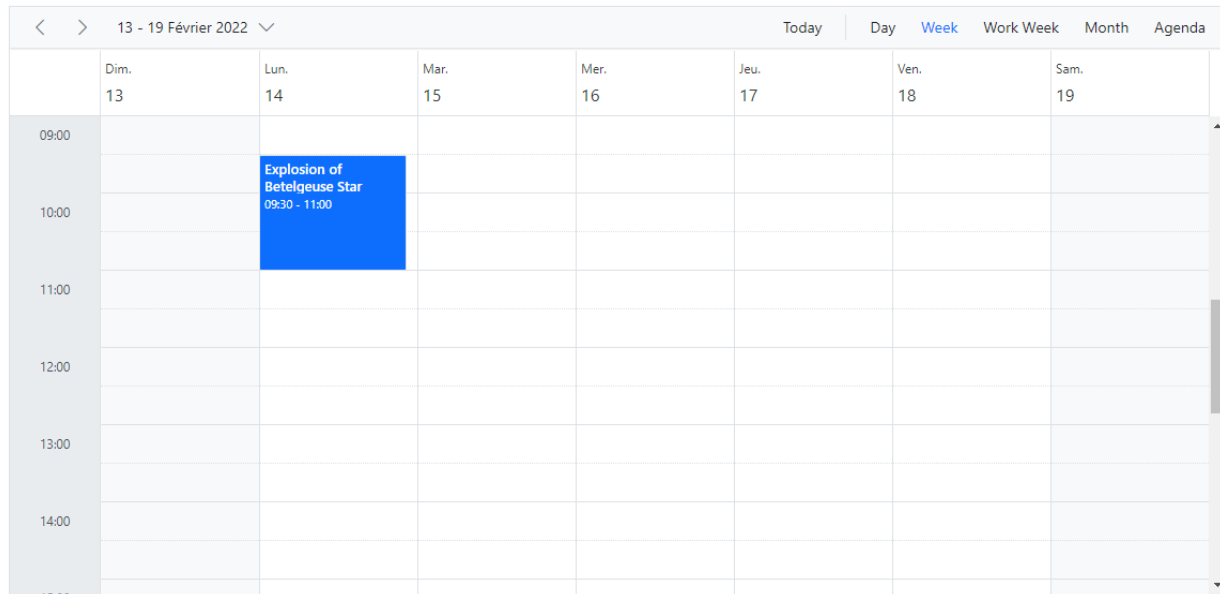
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
.Width("100%")
.Height("550px")
.Locale("fr-CH")
.EventSettings(new ScheduleEventSettings { DataSource = ViewBag.datasource
})
.SelectedDate(new DateTime(2022, 2, 15))
.Render()
)
<script>
loadCultureFiles('fr-CH');
function loadCultureFiles(name) {
var files = ['ca-gregorian.json', 'numberingSystems.json', 'numbers.json',
'timeZoneNames.json', 'ca-islamic.json'];
var loader = ej.base.loadCldr;
var loadCulture = function (prop) {
var val, ajax;
if (files[prop] === 'numberingSystems.json') {
ajax = new ej.base.Ajax(location.origin + '/../Content/cldr-
data/supplemental/' + files[prop], 'GET', false);
} else {
ajax = new ej.base.Ajax(location.origin + '/../Content/cldr-data/main/' +
name + '/' + files[prop], 'GET', false);
}

```

```
}
ajax.onSuccess = function (value) {
    val = value;
};
ajax.send();
loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
}
}
</script>
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    ViewBag.datasource = GetScheduleData();
    return View();
}
public List<AppointmentData> GetScheduleData()
{
    List<AppointmentData> appData = new List<AppointmentData>();
    appData.Add(new AppointmentData
    { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
    DateTime(2022, 2, 14, 9, 30, 0), EndTime = new DateTime(2022, 2, 14, 11, 0,
    0) });
    return appData;
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
}
```

Note: [View sample in GitHub](#). Refer this documentation to [localizing the static scheduler text](#)

Changing Global Culture and Currency Code

To set the default culture and the currencyCode for all ASP.NET MVC controls, you can use the methods `setCulture` for setting default locale and `setCurrencyCode` for setting the currencyCode in view page.

Setting Global Culture

CSHTML

```
<script>
ej.base.setCulture('ar');
</script>
```

Setting Currency Code

CSHTML

```
<script>
ej.base.setCurrencyCode('EUR');
</script>
```

Note: If global culture is not set, then `en-US` is set as default locale and `USD` is set as default currency code.

Manipulating Numbers

<!-- markdownlint-disable MD024 -->

Supported Format String

<!-- markdownlint-disable MD024 -->

Based on the `NumberFormatOptions` number formatting and parsing operations are processed. You need to specify some or all of the following properties mentioned in the below table.

No	Properties	Description
----	------------	-------------

| --- | --- | --- |

| 1 | **format** | Denotes the format to be set. Possible values are
 1. **N** - denotes numeric type.
 2. **C** - denotes currency type.
 3. **P** - denotes percentage type.
 E.g:
 `formatNumber(1234344,{format:'N4'})`.

 > If no format is specified it takes numeric as default format type. |

| 2 | **minimumFractionDigits** | Indicates the minimum number of fraction digits . Possible values are 0 to 20. |

| 3 | **maximumFractionDigits** | Indicates the maximum number of fraction digits. Possible values are 0 to 20. |

| 4 | **minimumSignificantDigits** | Indicates the minimum number of significant digits. Possible values are 1 to 21.
 > If **minimumSignificantDigits** is given it is mandatory to give **maximumSignificantDigits** |

| 5 | **maximumSignificantDigits** | Indicates the maximum number of significant digits. . Possible values are 1 to 21.
 > If **maximumSignificantDigits** is given it is mandatory to give **minimumSignificantDigits** |

| 6 | **useGrouping** | Indicates whether to enable the group separator or not. By default grouping value will be true. |

| 7 | **minimumIntegerDigits** | Indicates the minimum number of the integer digits to be placed in the value. Possible values are 1 to 21. |

| 8 | **currency** | Indicates the currency code which needs to be considered for the currency formatting. |

Note: The **minimumIntegerDigits**, **minimumFractionDigits** and **maximumFractionDigits** are categorized as group one, **minimumSignificantDigits** and **maximumSignificantDigits** are categorized as group two. If group two properties are defined, then the group one properties will be ignored.

Custom number formatting and parsing

Custom number formatting and parsing are also supported by specifying the pattern directly in the **format** property of **NumberFormatOptions**. Custom number format can be achieved by using one or more custom format specifiers listed in the below table.

Specifier	Description	Input	Format	Output
-----	-----	-----	-----	-----
0	Replaces the zero with the corresponding digit if one is present. Otherwise, zero appears in the result string.	<code>instance.formatNumber(123,{format: '0000' })</code>		'0123'
#	Replaces the "#" symbol with the corresponding digit if one is present; otherwise, no digit appears in the result string.	<code>instance.formatNumber(1234,{format: '####' })</code>		'1234'
.	Denotes the number of digits allowed after the decimal points if it's not specified then no need to specify decimal point values.	<code>instance.formatNumber(546321,{format: '###0.##0#' })</code>		'546321.000'
%	Percent specifier denotes the percentage type format.	<code>instance.formatNumber(1,{format: '0000 %' })</code>		'0100 %'

| \$ | Denotes the currency type format based on the global currency code specified. |

instance.formatNumber(13,{format: '\$ ###.00' }); | '\$ 13.00' |

| ; | Denotes separate formats for positive, negative and zero values. | instance.formatNumber(-120,{format: '###.##;(###.00);-0'}); | '(120.00)' |

| 'String' (single Quotes) | Denotes the characters enclosed within single Quote(') to be replaced in the resultant string. | instance.formatNumber(-123.44,{format: "####.## '@"}); | '123.44 @' |

Note: If custom format pattern is specified other `NumberFormatOptions` properties will not be considered.

Number formatting

`getNumberFormat`

The `getNumberFormat` method will return a function that formats given number based on the `NumberFormatOptions` specified.

CSHTML

```
<div>Value:<span class='format text'>1234545.65</span></div>
<div>Formatted Value:<span class='result text'></span></div>
<script>
var intl = new ej.base.Internationalization();
var nFormatter = intl.getNumberFormat({ skeleton: 'C3', currency:
'USD',minimumIntegerDigits:8});
var formattedValue = nFormatter(1234545.65)
document.querySelector('.result').innerHTML = formattedValue;
</script>
```

Value:1234545.65

Formatted Value:01,234,545.65

`formatNumber`

The `formatNumber` method which takes two arguments numeric value and `NumberFormatOptions`, returns the formatted string.

CSHTML

```
<div>Value:<span class='format text'>12345.65</span></div>
<div>Formatted Value:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var formattedString = intl.formatNumber(12345.65, { format:'C5' ,
useGrouping: false,
minimumSignificantDigits:1, maximumSignificantDigits:3 });
document.querySelector('.result').innerHTML = formattedString;
</script>
```

Value:12345.65
Formatted Value:\$12300

Parsing

`getNumberParser`

The `getNumberParser` method will return a function that parses given string based on the `NumberFormatOptions` specified.

CSHTML

```
<div>FormattedValue:<span class='format text'>123567.45%</span></div>
<div>ParsedOutput:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var nParser = intl.getNumberParser({ format:'P2' , useGrouping: false});
var val = nParser('123567.45%');
document.querySelector('.result').innerHTML = val + '';
</script>
```

FormattedValue:123567.45%
ParsedOutput:1235.67

`parseNumber`

The `parseNumber` method takes two arguments the string value, `NumberFormatOptions` and returns the numeric value.

CSHTML

```
<div>FormattedValue:<span class='format text'>$01,234,545.650</span></div>
<div>ParsedOutput:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var val = intl.parseNumber('$01,234,545.650', { format: 'C3', currency:
'USD', minimumIntegerDigits: 8 });
document.querySelector('.result').innerHTML = val + '';
</script>
```

FormattedValue:\$01,234,545.650
ParsedOutput:1234545.65

Manipulating DateTime

Supported Format String

Based on the `DateFormatOptions`, date formatting and parsing operations are processed. You need to specify some or all of the following properties mentioned below table.

Options	Descriptions
---	---
Type	It specifies the type of format to be used supported types . 1. date 2. dateTime 3. time Based on the type specified the supported skeletons are given below. 1. short 2. medium, 3. long 4. full E.g: <code>formatDate(new Date(), {type: 'date', skeleton:medium})</code> > If no type is specified then date type is set by default.
skeleton	Specifies the format in which the <code>dateTime</code> format will process

<!-- markdownlint-disable MD036 -->

Date type skeletons

skeleton	Option input	Format Output
---	---	---
short	{type: 'date', skeleton:'short'}	11/4/16
medium	{type: 'date', skeleton:'medium'}	Nov 4, 2016
long	{type: 'date', skeleton:'long'}	November 4, 2016
full	{type: 'date', skeleton:'full'}	Friday, November 4, 2016

Time type skeletons

skeleton	Option input	Format Output
---	---	---
short	{type: 'time', skeleton:'short'}	1:03 PM
medium	{type: 'time', skeleton:'medium'}	1:03:04 PM
Long	{type: 'time', skeleton:'long'}	1:03:04 PM GMT+5
full	{type: 'time', skeleton:'full'}	1:03:04 PM GMT+05:30

DateTime type skeletons

Skeleton	Option input	Format Output
---	---	---
short	{type: 'dateTime', skeleton:'short'}	11/4/16, 1:03 PM
medium	{type: 'dateTime', skeleton:'medium'}	Nov 4, 2016, 1:03:04 PM
Long	{type: 'dateTime', skeleton:'long'}	November 4, 2016 at 1:03:04 PM GMT+5
full	{type: 'dateTime', skeleton:'full'}	Friday, November 4, 2016 at 1:03:04 PM GMT+05:30

Additional skeletons

Apart from the standard date type formats additional format are supported by using the additional skeletons given in below table.

skeleton	Option input	Format Output
---	---	---
d	{skeleton:'d'}	7
E	{skeleton:'E'}	Mon
Ed	{skeleton:'Ed'}	7 Mon
Ehm	{skeleton:'Ehm'}	Mon 12:43 AM
EHm	{skeleton:'EHm;'}}}	Mon 12:43
Ehms	{skeleton:'Ehms' }	Mon 2:45:23 PM
EHms	{skeleton:'EHms'}}}	Mon 12:45:45
Gy	{skeleton:'Gy' }	2016 AD
GyMMM	{skeleton:'GyMMM'}	: Nov 2016 AD
GyMMMd	{skeleton:'GyMMMd'}	Nov 7, 2016 AD
GyMMMEd	{skeleton:'GyMMMEd'}	Mon, Nov 7, 2016 AD
h	{skeleton:'h'}	12 PM
H	{skeleton:'H'}	12
hm	{skeleton:'hm'}	12:59 PM
Hm	{skeleton:'Hm'}	12:59
hms	{skeleton:'hms'}	12:59:13 PM
Hms	{skeleton:'Hms'}	12:59:13
M	{skeleton:'M'}	11
Md	{skeleton:'Md'}	11/7
MEd	{skeleton:'hms'}	Mon, 11/7
MMM	{skeleton:'MMM'}	Nov
MMMEd	{skeleton:'MMMEd'}	Mon, Nov 7
MMMd	{skeleton:'MMMEd'}	Nov 7
ms	{skeleton:'ms'}	59:13
y	{skeleton:'y' }	2016
yM	{skeleton:'yM' }	11/2016
yMd	{skeleton:'yMd' }	11/7/2016
yMEd	{skeleton:'yMEd' }	Mon, 11/7/2016
yMMM	{skeleton:'yMMM' }	Nov 2016

yMMMd	{skeleton:'yMMMd'}	Nov 7, 2016
yMMMEd	{skeleton:'yMMMEd'}	Mon, Nov 7, 2016
yMMM	{skeleton:'yMMM'}	Nov 2016

Note: Culture specific format skeletons are also supported.

Custom Formats

To use the custom date and time formats, specify the date/time pattern directly in the *format* property. Custom format string must contain one or more of the following standard date/time symbols

Symbols	Description
-----	-----
G	Denotes the era in the date
y	Denotes the year.
M / L	Denotes month.
E / c	Denotes the day of week.
d	Denotes the day of month.
h / H	Denotes the hour. <i>h</i> for 12 hour and <i>H</i> for 24 hours format.
m	Denotes minutes.
s	Denotes seconds.
a	Denotes the am/pm designator it will only be displayed if hour is specified in the <i>h</i> format.
z	Denotes the time zone.
' (single quotes)	To display words in the formatted date you can specify the words with in the single quotes

Custom format example

CSHTML

```
<div>DateValue:<span class='format text'>new Date('1/12/2014
10:20:33')</span></div>
<div>Formatted Value:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var formattedString = intl.formatDate(new Date('1/12/2014 10:20:33'), {
format: '\'year:\'y\' + " " + \''month:\' MM\' }');
document.querySelector('.result').innerHTML = formattedString;
</script>
```

DateValue:new Date('1/12/2014 10:20:33')

Formatted Value:year:2014 month: 01

Note: If format property is given in options other properties are not considered.

```
<!-- markdownlint-enable MD036 -->
```

Formatting

`getDateFormat`

The `getDateFormat` method which will return a function that formats given date object based on the `DateFormatOptions` specified.

CSHTML

```
<div>DateValue:<span class='format text'>new Date('1/12/2014
10:20:33')</span></div>
<div>Formatted Value:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var dFormatter = intl.getDateFormat({ skeleton: 'full', type: 'dateTime' });
var formattedString = dFormatter(new Date('1/12/2014 10:20:33'));
document.querySelector('.result').innerHTML = formattedString;
</script>
```

DateValue:**new Date('1/12/2014 10:20:33')**
Formatted Value:**Sunday, January 12, 2014 at 10:20:33 AM GMT+05:30**

`formatDate`

The `formatDate` method takes two arguments date object and `DateFormatOptions`, and returns the formatted string.

CSHTML

```
<div>DateValue:<span class='format text'>new Date('1/12/2014
10:20:33')</span></div>
<div>Formatted Value:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var date = new Date();
var formattedString = intl.formatDate(new Date('1/12/2014 10:20:33'), {
skeleton: 'GyMMM' });
document.querySelector('.result').innerHTML = formattedString;
</script>
```

DateValue:**new Date('1/12/2014 10:20:33')**
Formatted Value:**Jan 2014 AD**

```
<!-- markdownlint-enable MD036 -->
```


*Parsing**`getDateParser`*

The `getDateParser` method will return a function that parses given string based on the `DateFormatOptions` specified.

CSHTML

```
<div>Formatted value:<span class='format text'>Friday, November 4, 2016 at 1:03:04 PM GMT+05:30</span></div>
<div>parsed Value:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var dParser = intl.getDateParser({skeleton: 'full', type: 'dateTime'});
var val = dParser('Friday, November 4, 2016 at 1:03:04 PM GMT+05:30');
document.querySelector('.result').innerHTML = val.toString();
</script>
```

Formatted value:Friday, November 4, 2016 at 1:03:04 PM GMT+05:30
 parsed Value:Fri Nov 04 2016 13:03:04 GMT+0530 (India Standard Time)

`parseDate`

The `parseDate` method takes two arguments string value, `DateFormatOptions` and returns the date Object.

CSHTML

```
<div>Formatted value:<span class='format text'>11/2016</span></div>
<div>parsed Value:<span class='result text'> </span></div>
<script>
var intl = new ej.base.Internationalization();
var val = intl.parseDate('11/2016',{skeleton: 'yM'});
document.querySelector('.result').innerHTML = val.toString();
</script>
```

Formatted value:11/2016
 parsed Value:Tue Nov 01 2016 00:00:00 GMT+0530 (India Standard Time)

Right-To-Left support in Syncfusion ASP.NET MVC controls

The right-to-left (RTL) support can be enabled for Syncfusion ASP.NET MVC controls by setting `enableRtl` property to `true`. This will render all the Syncfusion ASP.NET MVC controls in the right-to-left direction.

Enable RTL for all controls

You can enable right to left (RTL) for all Syncfusion controls used in the application by setting `enableRtl` property to `true` in the script tag. Add the below code snippet in the `_Layout.cshtml` file inside `body` tag.

~/ _LAYOUT.CSHTML

```
<script>
// Enables Right to left alignment for all controls
ej.base.enableRtl(true);
</script>
```

Drag a column header here to group its column				
Ship Country	Freight	Employee ID	Customer ID	Order ID
Denmark	\$2.30	1	ALFKI	10001
Brazil	\$3.30	3	ANATR	10002
Germany	\$4.30	2	ANTON	10003
Austria	\$5.30	4	BLONP	10004
Switzerland	\$6.30	5	BOLID	10005
Denmark	\$4.60	2	ALFKI	10006
Brazil	\$6.60	4	ANATR	10007
Germany	\$8.60	3	ANTON	10008
Austria	\$10.60	5	BLONP	10009
Switzerland	\$12.60	6	BOLID	10010
Denmark	\$6.90	3	ALFKI	10011
Brazil	\$9.90	5	ANATR	10012

of 4 pages (45 items) 1

Enable RTL to individual control

To control a control's direction individually, you can directly set the control's `enableRtl` property as `true`. For illustration, RTL has been enabled for Schedule control in following code snippet.

RTL.CS

```
@Html.EJS().Schedule("schedule").Render()
```

AGENDA	MONTH	WORK WEEK	WEEK	DAY	TODAY	February 20 - 26, 2022		<	>
	Sat 26	Fri 25	Thu 24	Wed 23	Tue 22	Mon 21	Sun 20		
								AM 9:00	
								AM 10:00	
								AM 11:00	
								PM 12:00	
								PM 1:00	
								PM 2:00	

State Persistence in ASP.NET MVC

The Syncfusion ASP.NET MVC library supports persisting a control's state across page refreshes or navigation. To enable this feature, set `enablePersistence` property as true to the required control. This will store the control's state in browser's `localStorage` object on page `unload` event.

CSHTML

```
@Html.EJS().Grid("check").DataSource(dataManger =>
{
    dataManger.Url("http://services.odata.org/V4/Northwind/Northwind.svc/Product
s").CrossDomain(true).Adaptor("ODataV4Adaptor");
}).Columns(col =>
{
    col.Field("ProductID").HeaderText("Product
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ProductName").HeaderText("Product
Name").Width("150").Add();
    col.Field("UnitPrice").HeaderText("Supplier
ID").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("UnitsInStock").HeaderText("QuantityPerUnit").Width("120").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Discontinued").HeaderText("Discontinued").Width("140").TextAlign(
Syncfusion.EJ2.Grids.TextAlign.Center).Type("boolean").DisplayAsCheckBox(tru
e).Add();
})
.EnablePersistence()
.AllowPaging()
.PageSettings(page=>page.PageCount(5))
.Render()
```

State Persistence Supported controls and Properties

The following table demonstrates the list of Syncfusion ASP.NET MVC controls that are supported with state persistence and describes the list of properties stored in the `localStorage`.

<!-- markdownlint-disable MD033 -->

control Name	Properties
<u>Grid</u>	<ul style="list-style-type: none"> Columns filterSettings searchSettings groupSettings pageSettings selectedRowIndex scrollPosition
Accordion	<ul style="list-style-type: none"> expandedIndices
<u>Tabs</u>	<ul style="list-style-type: none"> selectedItem
<u>Schedule</u>	<ul style="list-style-type: none"> currentView selectedDate scrollLeft scrollTop
<u>Kanban</u>	<ul style="list-style-type: none"> columns dataSource swimlaneToggleArray
Chart	<ul style="list-style-type: none"> zoomFactor zoomPosition
<u>Maps</u>	<ul style="list-style-type: none"> zoomSettings
<u>Pivot Table</u>	<ul style="list-style-type: none"> dataSourceSettings pivotValues gridSettings chartSettings displayOption
<u>TreeGrid</u>	<ul style="list-style-type: none"> columns pageSettings searchSettings filterSettings selectedRowIndex sortSettings
Switch	<ul style="list-style-type: none"> checked

Checkbox	<ul style="list-style-type: none">checked
RadioButton	<ul style="list-style-type: none">checkedvalue
ColorPicker	<ul style="list-style-type: none">value
ListBox	<ul style="list-style-type: none">value
<u>QueryBuilder</u>	<ul style="list-style-type: none">rule
In-placeEditor	<ul style="list-style-type: none">value
RichTextEditor	<ul style="list-style-type: none">value
Splitter	<ul style="list-style-type: none">paneSettings
Autocomplete	<ul style="list-style-type: none">value
Calendar	<ul style="list-style-type: none">value
ComboBox	<ul style="list-style-type: none">value
DatePicker	<ul style="list-style-type: none">value
DropDownList	<ul style="list-style-type: none">value
MaskedTextBox	<ul style="list-style-type: none">value
NumericTextBox	<ul style="list-style-type: none">value
Textbox	<ul style="list-style-type: none">value
TimePicker	<ul style="list-style-type: none">value
Multiselect	<ul style="list-style-type: none">value
DateTimePicker	<ul style="list-style-type: none">value
DateRangePicker	<ul style="list-style-type: none">startDateendDatevalue

Uploader	<ul style="list-style-type: none"> filesData
Slider	<ul style="list-style-type: none"> value
ListView	<ul style="list-style-type: none"> cssClass enableRtl htmlAttributes enable fields animation headerTitle sortOrder showIcon height width showCheckBox checkBoxPosition
TreeView	<ul style="list-style-type: none"> selectedNodes checkedNodes expandedNodes
Dashboard Layout	<ul style="list-style-type: none"> panels
File Manager	<ul style="list-style-type: none"> view path selectedItems
Sidebar	<ul style="list-style-type: none"> type position isOpen
Dropdown Tree	<ul style="list-style-type: none"> value

<!-- markdownlint-enable MD033 -->

Animation in Syncfusion ASP.NET MVC

The **Animation** library is used to perform animation effects on HTML elements by running sequence of frames.

Animation Supported controls

The following list demonstrates the Syncfusion ASP.NET MVC control documents that are supported with Animation.

- [Accordion](#)

- [Accumulation Chart](#)
- [Bullet Chart](#)
- [Chart](#)
- [Circular Gauge](#)
- [Context Menu](#)
- [Dialog](#)
- [In-place Editor](#)
- [ProgressBar](#)
- [SideBar](#)
- [Tabs](#)
- [Toast](#)
- [ToolTip](#)
- ListView
- Menu
- Pivot Table
- ProgressButton
- Smith Chart
- Stock Chart
- TreeView

Animating a HTML Element

The `animate` method of Animation library can be used to animate the HTML elements. This method can also take additional `AnimationModel`.

CSHTML

```
<div id="fade"></div>
<div id="zoom"></div>
<script>
var animation = new ej.base.Animation({ duration: 5000 });
animation.animate('#fade', { name: 'FadeOut' });
animation.animate('#zoom', { name: 'ZoomOut' });
</script>
<style>
#fade, #zoom {
background: #333333;
border: 1px solid #cecece;
box-sizing: border-box;
float: left;
height: 100px;
width: 100px;
}
</style>
```



Note: [View sample in GitHub](#)

Enable or disable animation globally

Enable or disable animation for all ASP.Net MVC controls globally by using the `setGlobalAnimation` method with one of the below options:

- `GlobalAnimationMode.Enable` - Enables the animation for all components, regardless of the individual component's animation settings.
- `GlobalAnimationMode.Disable` - Disables the animation for all components, regardless of the individual component's animation settings.
- `GlobalAnimationMode.Default` - Animation is enabled or disabled based on the component's animation settings.

In the below code snippet, animation is disabled.

~/ LAYOUT.CSHTML

```
<script>
ej.base.setGlobalAnimation(ej.base.GlobalAnimationMode.Disable);
</script>
```

`setGlobalAnimation` method controls script-level animations only, and it is not applicable for direct CSS-level animations (animations defined from CSS classes or properties).

Drag and Drop support in Syncfusion ASP.NET MVC controls

- Drag and Drop support can be enabled for Syncfusion ASP.NET MVC controls by setting `allowDragAndDrop` property to `true`. It allows to drag and drop the specific elements in the Syncfusion ASP.NET MVC controls.
- Drag and Drop is supported through two libraries of Essential JS 2. Those are [Draggable](#) and [Droppable](#). Draggable makes DOM to be dragged using mouse or touch gestures and Droppable mark required DOM as droppable zone.

Drag and Drop Supported controls

The following list demonstrates the Syncfusion ASP.NET MVC control documents that are supported with Drag and Drop.

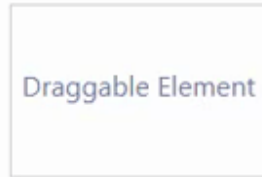
- [Tabs](#)
- [Dashboard Layout](#)
- [Schedule](#)
- [File Manager](#)
- [Gantt](#)
- [Grid](#)
- [Kanban](#)
- [ListBox](#)
- [ListView](#)
- [TreeGrid](#)
- [TreeView](#)

Initializing custom Draggable element

You can make any element draggable by passing the element to Draggable constructor.

CSHTML

```
<div id="drag-element"><p>Draggable Element </p></div>
<script>
var dragElement = document.getElementById('drag-element');
var draggable = new ej.base.Draggable(dragElement, {clone: false});
</script>
<style>
#drag-element {
height: 100px;
width: 150px;
border: 1px solid #cecece;
cursor: move;
user-select: none;
color: #6a77a7;
touch-action: none;
}
p {
padding-top: 20px;
text-align: center;
margin: 14px 0px 14px 0px;
}
</style>
```

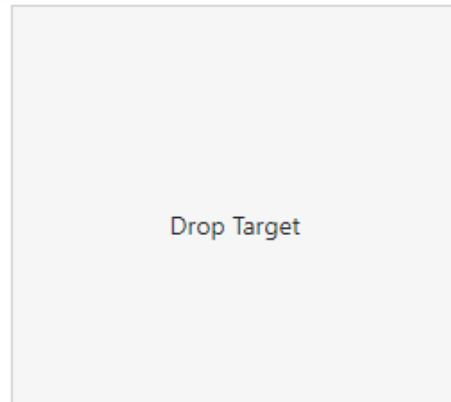


Creating Droppable zone

You can convert any DOM element as a droppable zone, which accepts the draggable elements.

CSHTML

```
<div id="droppable">
  <p class="drop">
    <span>Drop Target</span>
  </p>
</div>
<script>
var droppable = new ej.base.Droppable(document.getElementById('droppable'));
</script>
<style>
#droppable {
margin: 5px;
line-height: 170px;
font-size: 14px;
width: 250px;
border: 1px solid #cecece;
background: #f6f6f6;
touch-action: none;
}
.drop {
padding-top: 23px;
text-align: center;
margin: 14px 0px 14px 0px;
}
</style>
```



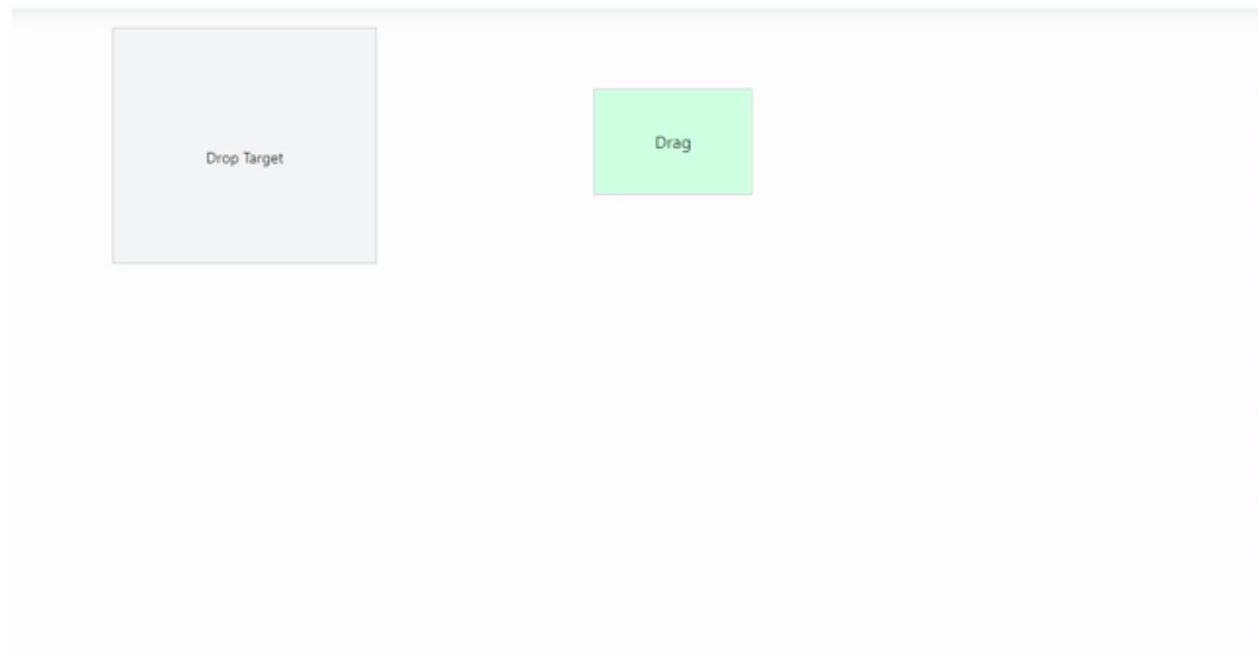
Defining Drop Action

To define drop action set [drop](#) callback function during droppable object creation. You can get details of dropped element through dropped element property in event argument.

CSHTML

```
<div id="droppable"><p class="drop"><span>Drop Target </span></p></div>
<div id="drag-element"><p class="drag-text">Drag </p></div>
<script>
var draggable = new ej.base.Draggable(document.getElementById('drag-
element'), {clone: false});
var droppable = new ej.base.Droppable(document.getElementById('droppable'),
{
drop: (function (e) {
e.droppedElement.querySelector('.drag-text').textContent = 'Dropped';
})
});
</script>
<style>
#drag-element {
height: 100px;
width: 150px;
border: 1px solid #cecece;
cursor: move;
background: #cdffe3;
user-select: none;
touch-action: none;
}
#droppable {
margin: 5px;
line-height: 170px;
font-size: 14px;
width: 250px;
border: 1px solid #cecece;
background: #f6f6f6;
}
```

```
touch-action: none;
}
.drop, .drag-text {
padding-top: 23px;
text-align: center;
margin: 14px 0px 14px 0px;
}
</style>
```



Note: [View sample in GitHub](#)

See also

- [Define handle element for Draggable](#)
- [Restricting Draggable within container](#)
- [Visual feedback of draggable element](#)
- [Accepting specific drag element in droppable](#)

Compatibility with Syncfusion ASP.NET MVC (Essential JS 1)

This documentation explains how to render Essential JS 1 and Essential JS 2 Syncfusion components in a single page.

Adding application

You can create Essential JS 1 and Essential JS 2 controls using the following links:

Getting started for [EJ2 ASPMVC control](#)

Getting started for [EJ1 ASPMVC control](#)

Style compatibility

Add compatibility styles. Essential JS 1 and Essential JS 2 controls have compatibility styles that do not affect each other.

```
`html
<head>
@ Syncfusion Essential JS 1 Styles @
@Styles.Render("https://cdn.syncfusion.com/16.1.0.24/js/web/bootstrap-
theme/ej.web.all.compatibility.min.css")
@ Syncfusion Essential JS 2 Styles @
@Styles.Render("https://cdn.syncfusion.com/ej2/styles/compatibility/material.css")
</head>
`
```

Script compatibility

Add scripts for Essential JS 1 and Essential JS 2 as shown in the following codes

```
`html
@ Syncfusion Essential JS 1 Scripts @
<script src="http://cdn.syncfusion.com/js/assets/external/jquery3.3.1.min.js"></script>
<script src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></script>
<script src="http://cdn.syncfusion.com/16.3.0.21/js/web/ej.web.all.min.js"></script>
@ Syncfusion Essential JS 2 Scripts @
<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
`
```

Note: If you do not add ej1 script before ej2 script, it will throw script error.

Adding compatibility

To add compatibility, use the following code in the `_Layout.cshtml` page. Since EJ1 and EJ2 controls have same library names to perform different actions, conflicts may occur when you refer these both controls in same application. To overcome this, extend these libraries in ej namespace.

```
`html
<script>
var dataCopy = Object.assign({}, ej.data);
$.extend(ej, Syncfusion);
$.extend(ej.data, dataCopy);
</script>
`
```

Adding script manager

Define script manager for both Essential JS 1 and Essential JS 2:

```
`html
@Html.EJ().ScriptManager()
```

```
@Html.EJS().ScriptManager()
```

```
,
```

The following code illustrates the final view of layout page.

```
`html
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
@ Syncfusion Essential JS 1 Styles @
@Styles.Render("http://cdn.syncfusion.com/16.3.0.21/js/web/flat-azure/ej.web.all.min.css")
@ Syncfusion Essential JS 2 Styles @
@Styles.Render("https://cdn.syncfusion.com/ej2/styles/compatibility/material.css")
</head>
<body>
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@Scripts.Render("~/Scripts/jsrender.min.js")
@ Syncfusion Essential JS 1 Scripts @
@Scripts.Render("~/Scripts/ej/web/ej.web.all.min.js")
@ Syncfusion Essential JS 2 Scripts @
@Scripts.Render("~/Scripts/ej2/ej2.min.js")
@RenderSection("scripts", required: false)
<script>
var dataCopy = Object.assign({}, ej.data);
$.extend(ej, Syncfusion);
$.extend(ej.data, dataCopy);
</script>
@Html.EJ().ScriptManager()
@Html.EJS().ScriptManager()
<div class="container body-content">
@RenderBody()
</div>
</body>
</html>
```

Declare Essential JS 1 and Essential JS 2 controls

Essential JS 1 and Essential JS 2 controls have already been initialized via getting started. If you want change control, you can initialize using the following code.

```
`html
<h2> Essential JS 1 - Grid Control</h2>
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.gridData)
.AllowPaging().PageSettings(page => page.PageSize(4))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("CustomerID").Width(80).Add();
col.Field("ShipName").HeaderText("ShipName").Width(100).Add();
col.Field("ShipCity").HeaderText("ShipCity").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAlign(TextAlign.Right).Add();
})
)
<h2> Essential JS 2 - Grid Component </h2>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.gridData).Columns(col =>
{
col.Field("OrderID").HeaderText("OrderID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
col.Field("OrderDate").HeaderText("OrderDate").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().PageSettings(page => page.PageSize(4).PageSizes(true)).Render()
`
```

Note: [View Sample in GitHub](#).

Template Engine

Syncfusion ASP.NET MVC (Essential JS 2) has built-in template engine which provides options to compile template string into a executable function. Then the generated executable function can be used for rendering DOM element using desired data.

Compiling

`compile` method from `ej2-base` can be used to convert our template strings into executable functions.

CSHTML

```
<div id="element"></div>
<script>
  // data
  var data = { name: 'Aston Martin' };
  // Compiling template string into executable function
  var getDOMString = ej.base.compile('<div>${name}</div>');
  // Using generated function to get output element collection
  var output = getDOMString(data);
  // append html collection to element
  document.getElementById('element').appendChild(output[0]);
</script>
```

Available Template Syntax

Name	Syntax	Description
---	---	---
Expression	<div>\${name}</div>	We have expression evolution as like ES6 expression string literals.
Dot Variable Access	<div>\${person.info.name}</div>	Access the json variable with dot notation.
Variable Function	<div>\${name.toUpperCase()}</div>	Utilize the variable function example, name.toUpperCase()
Window Function	<div>\${getCurrentTime()}</div>	Use window function inside template. > Here, getCurrentTime() is a function that defined in the window object.
Custom Helper Function	<div>\${covertToCurrency()}</div>	Use function that passed in helper function.
IF Else Statement	<div> \${if(gender=== "male")} Male \${else} Female \${/if} </div>	Branching statement in Template.
For Statement	<div> \${for(mark of marks)} \${mark} \${/for} </div>	Use for looping inside template.
For Index value access	<div> \${for(mark of marks)} \${markIndex} \${/for} </div>	Get the index value of item while using for statement. Use the variable Index that suffixed with loop item name.

Custom Helper

Custom helper function can be defined and passed to `compile` function. Refer to the following example.

CSHTML

```
<div id="element"></div>
<script>
  var customHelper = {
    upperCase: function upperCase(str) {
      return str.toUpperCase();
    }
  };
</script>
```



```
    }  
    };  
    var data = { name: 'Aston Martin' };  
    var getDOMString = ej.base.compile('<div>${upperCase(name)}</div>',  
    customHelper);  
    let opElem = getDOMString(data);  
  
    document.getElementById('element').appendChild(opElem[0]);  
</script>
```

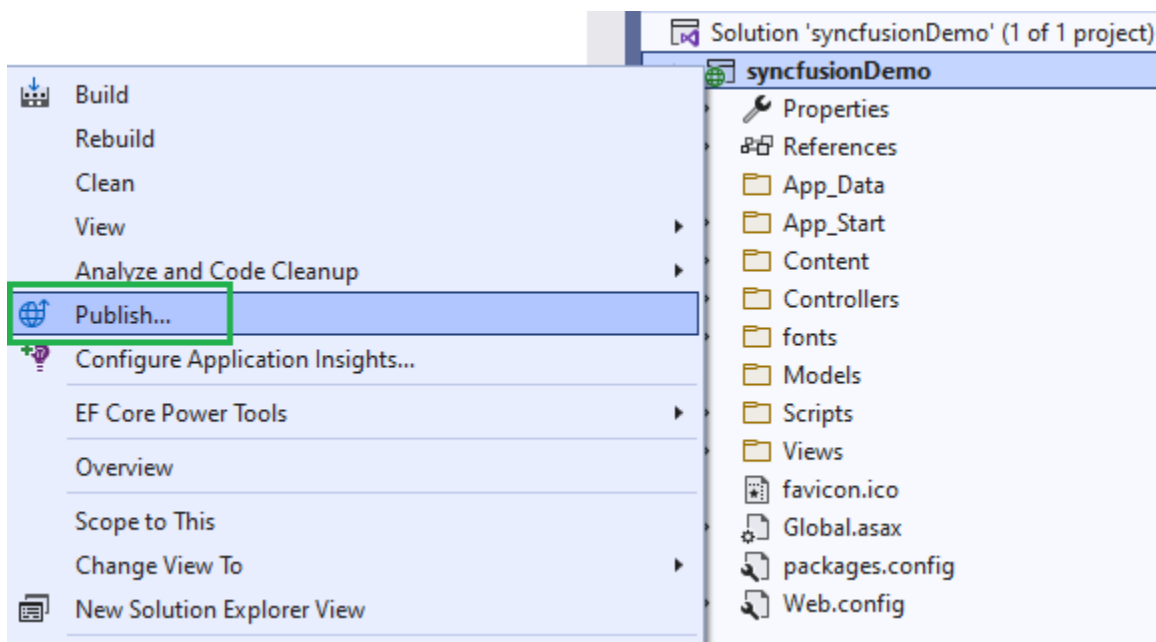
Deployment in ASP.NET MVC

This section provides information about deploying ASP.NET MVC applications with the Syncfusion ASP.NET MVC controls.

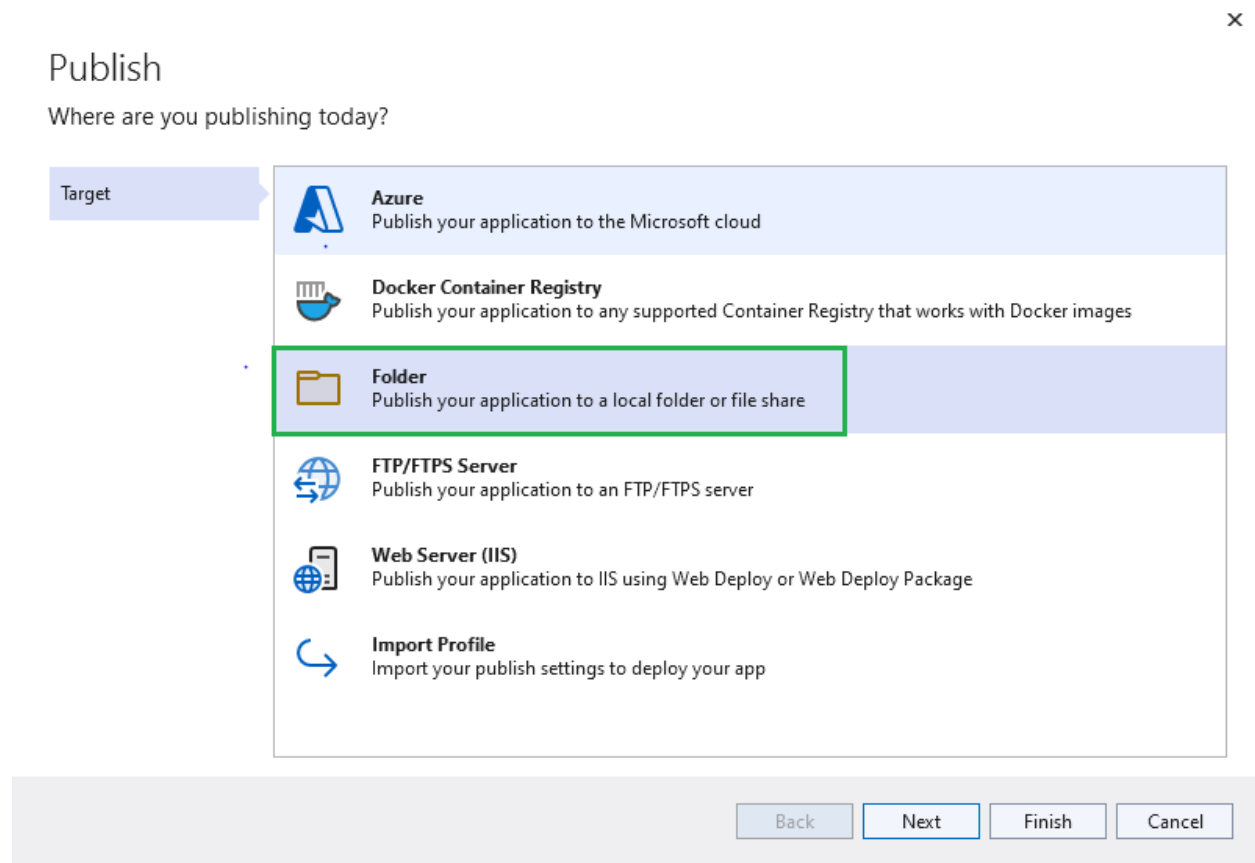
Refer to [ASP.NET MVC Deployment](#) topic for more information.

Publish ASP.NET MVC Application with Visual Studio

- Right-click on the project in the **Solution Explorer** and select **Publish**.



- Then, select the **Folder** option and click **Next** to select the publishing target location.



- Browse and specify the target location.

X

Publish

Provide the path to a local or network folder

Target

Folder location

D:\Development

Browse...

Location

For local folders you can provide either a full path or a relative path to the project, for example:

- publish\ (relative path)
- C:\Users\Username\Documents (full path)

For network folders you have to use \\ and then either the computer name or IP address, for example:

- \\server1\fileshare1
- \\192.168.1.17\fileshare1

Back Next Finish Cancel

- Then, click **Finish** and check the configuration as Release by clicking the edit option below the target location.

FolderProfile.pubxml Folder

Publish

+ New More actions

Ready to publish.

Settings

Target location	D:\Development
Delete existing files	False
Configuration	Release

Show all settings

- Then, click **Save** and **Publish**.

Note: Refer [here](#) for publishing the application to Azure App Service using Visual Studio.

Publish ASP.NET MVC Application with CLI

Packing the application and its dependencies into a folder for deployment to a hosting system by using the **MSBuild** command.

Use the following command to specify the path for the output directory.

.NET CLI

```
msbuild <ProjectName> /p:Configuration=Release /p:outdir="<output directory>"
```

you can find the published folder in this `<output directory>/_PublishedWebsites/<ProjectName>` location.

Refer to the MSBuild [optional arguments](#).

If the path is relative, the output directory generated is relative to the project file location, not to the current working directory.

Now, you can host the published folder by using the IIS or Azure app service.

See also

- [Publish a Web app to Azure App Service using Visual Studio](#)
- [Publish a Web app to Azure App Service using Visual Studio for Mac](#)

Input Form Validation

The Syncfusion ASP.NET MVC UI input and editor controls can be validated by using [FormValidator](#) in client side.

How to Validate Syncfusion ASP.NET MVC UI Controls

1. Add the **Form** component in the **Index.cshtml** page. Also, declare the Syncfusion ASP.NET MVC [TextBox](#) control inside the **Form** component.

CSHTML

```
@{
    IDictionary<string, object> firstNameAttribute = new Dictionary<string,
    object>()
    {
        { "name", "FirstName" }
    };
    IDictionary<string, object> lastNameAttribute = new Dictionary<string,
    object>()
    {
        { "name", "LastName" }
    };
}
<form id="form-element">
<div class="form-group">
<label for="first-name">First Name:</label>
@Html.EJS().TextBox("firstname").Placeholder("First
Name").Width("25%").HtmlAttributes(firstNameAttribute).Render()
</div>
<div class="form-group">
```

```
<label for="last-name">Last Name:</label>
@Html.EJS().TextBox("lastname").Placeholder("Last
Name").Width("25%").HtmlAttributes(lastNameAttribute).Render()
</div>
@Html.EJS().Button("submit").Content("Submit").Render()
</form>
```

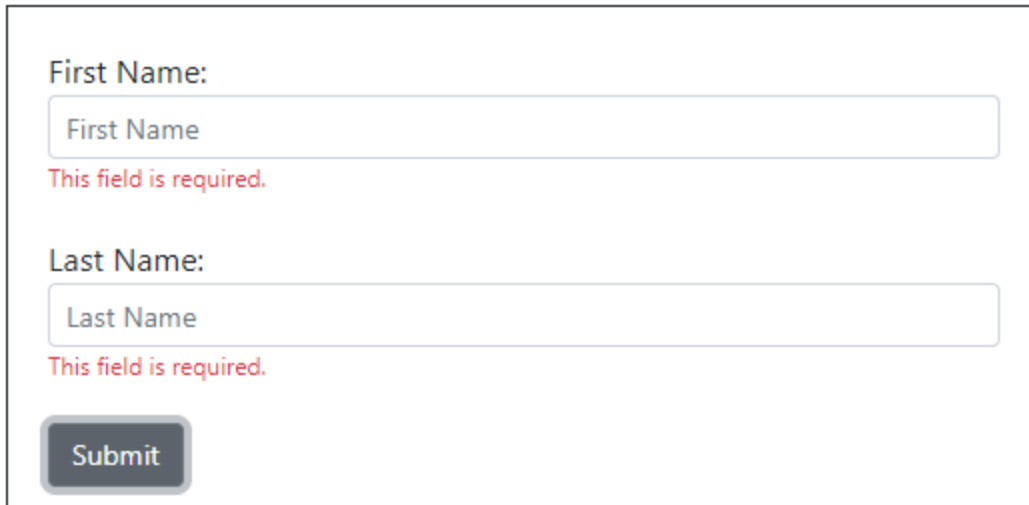
2.The following code shows how to perform client side validation using **FormValidator** inside the script tag.

CSHTML

```
<script>
// Sets required property in the FormValidator rules collection
var options = {
rules: {
'FirstName': { required: true },
'LastName': { required: true },
},
};
// Defines FormValidator to validate the TextBox
var formObject = new ej.inputs.FormValidator('#form-element', options);
// Places error label outside the TextBox using the customPlacement event of
FormValidator
formObject.customPlacement = function (element, errorElement) {
element.parentElement.parentElement.appendChild(errorElement);
};
// Form validates the input values using validate method of FormValidator
document.getElementById("submit").addEventListener('click', function () {
formObject.validate();
});
</script>
```

3.The **Form** validates the input values and displays the validation message on the **FormValidator** rules when submitting the form.

Validation Failure:



A form with two text input fields. The first field is labeled "First Name:" and contains the text "First Name". Below it is a red error message "This field is required.". The second field is labeled "Last Name:" and contains the text "Last Name". Below it is a red error message "This field is required.". At the bottom left is a dark gray "Submit" button.

Note: TextBox control supports three types of validation styles. Refer [here](#).

4.You can use the FormValidator [rules](#) property to display the custom validation error message for each input control.

CSHTML

```
<script>
// sets required property in the FormValidator rules collection
var options = {
rules: {
'FirstName': { required: [true, "Please enter FirstName" ] },
'LastName': { required: [true, "Please enter LastName" ] },
},
};
</script>
```



A form with two text input fields. The first field is labeled "First Name:" and contains the text "First Name". Below it is a red error message "Please enter FirstName". The second field is labeled "Last Name:" and contains the text "Last Name". Below it is a red error message "Please enter LastName". At the bottom left is a dark gray "Submit" button.

5.You can also use the [submit](#) event from FromValidator to validate the form manually.

ASP.NET MVC Form Validation Supported Controls

The following section provides the details about the Syncfusion ASP.NET MVC UI controls that are supported with form validation.

- [Masked TextBox](#)
- [NumericTextBox](#)
- [Range Slider](#)
- [CheckBox](#)
- [Radio Button](#)
- [Switch](#)
- [DatePicker](#)
- [TimePicker](#)
- [ListBox](#)
- [In-place Editor](#)
- [TextBox](#)
- [RichTextEditor](#)
- DateTimePicker
- Calendar
- ColorPicker
- DateRangePicker
- Auto Complete
- ComboBox
- DropDownList
- MultiSelect

See Also

- [ASP.NET MVC Forms and Validation](#)
- [Model validation in ASP.NET MVC and Razor Pages](#)

How To

How to troubleshoot compile-time and run-time errors

Compile-time error

Cannot find name 'Map' in ej2.d.ts: Need to change your target library. Try changing the 'lib' compiler option to es2015 or later.

You may see the below error while running the application.

Build:Cannot find name 'Map'. Do you need to change your target library? Try changing the 'lib' compiler option to es2015 or later.

Cause:

This error is thrown due to not including the required target library for the TypeScript compiler option and it can be resolved by any one of the below solutions.

Solutions:

1. **Using MS build**

By adding the required target `dom,es2015` library in `TypeScriptLib` MSBuild property in your `.csproj` file as like below, following the `TypeScriptToolsVersion` tag.

```
`cs
<TypeScriptToolsVersion>3.1</TypeScriptToolsVersion>
<TypeScriptLib>dom,es2015</TypeScriptLib>
`
```

Note: If `tsconfig.json` exists in your project, the compiler will prioritize using the specified configuration options from `tsconfig.json` file. Otherwise, it'll use the specified configuration options from the project file (`.csproj`).

Refer [KB Link](#) for more details.

2. Using `tsconfig.json`

By adding the required target library in `"compilerOptions"` property as like below.

```
`json
{
  "compilerOptions": {
    //...
    "target": "ES2015"
  },
  //...
}
```

Visual Studio Integration

Visual Studio Integration

Overview

The Syncfusion ASP.NET MVC (Essential JS 2) Visual Studio Extensions can be accessed through the Syncfusion Menu to create and configure the project with Syncfusion references in Visual Studio.

Note: Syncfusion Extension is published in the Visual Studio Marketplace. We provided Separate ASP.NET Core (Essential JS 2) Extension support for Visual Studio 2022 and Visual Studio 2019 or lower. Refer below Visual Marketplace link.

[Visual Studio 2022](#)

[Visual Studio 2019 or lower](#)

IMPORTANT

The Syncfusion ASP.NET MVC (Essential JS 2) menu option is available from v17.1.0.32.

Syncfusion provides the following supports in Visual Studio:

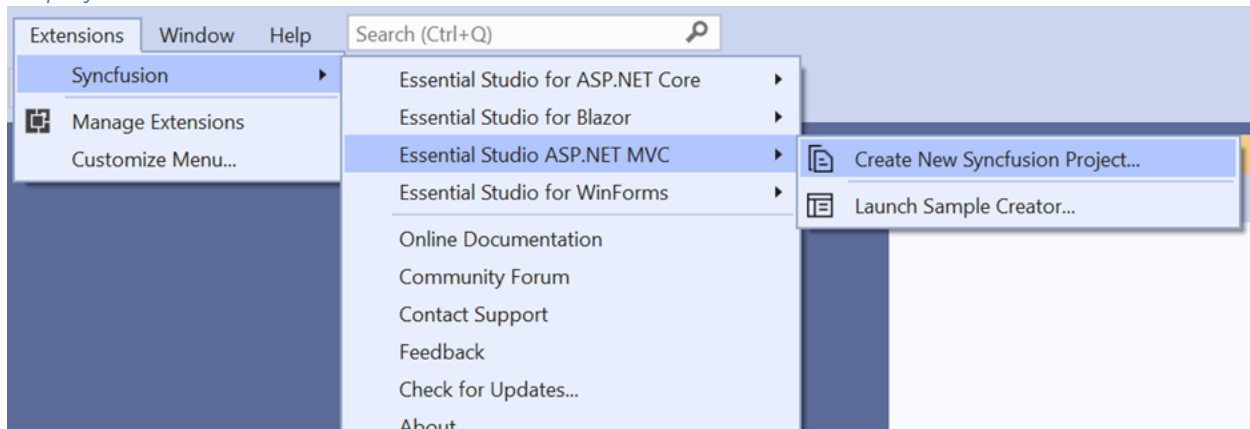
Project Template: Creates the Syncfusion ASP.NET MVC (Essential JS 2) application by adding the required Essential JS 2 components.

Convert Project: Converts an existing ASP.NET MVC application into a Syncfusion ASP.NET MVC (Essential JS 2) application by adding the required Syncfusion assemblies and resource files.

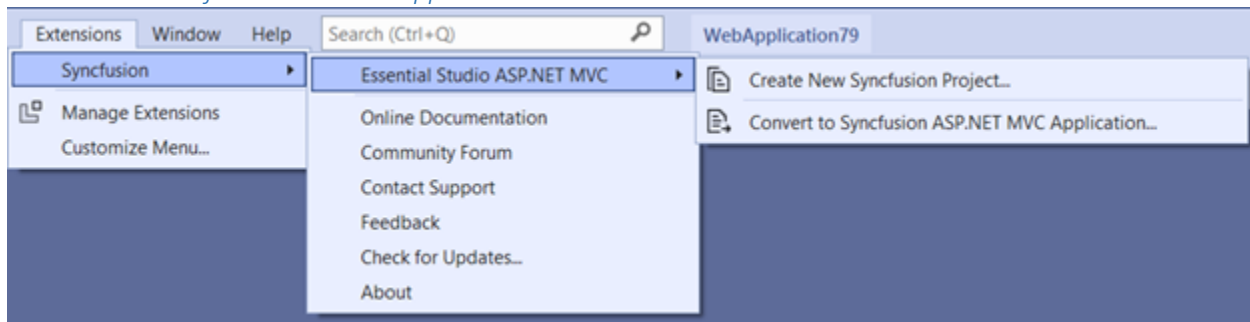
Upgrade Project: Upgrades the existing Syncfusion ASP.NET MVC (Essential JS 2) application from one Essential Studio version to another version.

Creator Sample: Creates the Syncfusion ASP.NET MVC (Essential JS2) application with the sample code of required controls and features.

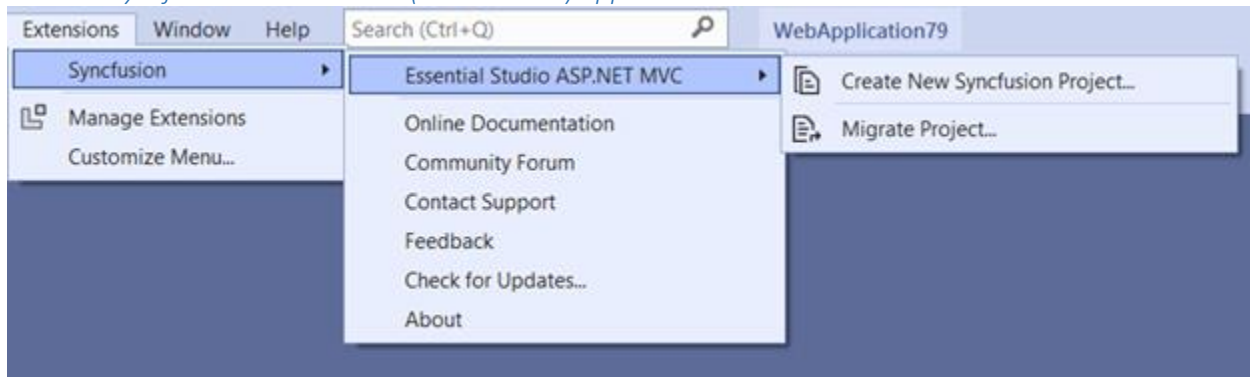
No project selected in Visual Studio



Selected Microsoft ASP.NET MVC application in Visual Studio



Selected Syncfusion ASP.NET MVC (Essential JS2) application in Visual Studio



Note: In Visual Studio 2017 or lower, you can see the Syncfusion menu directly in the Visual Studio menu.

Download and Installation

Syncfusion publishes the Visual Studio extension in the below Visual Studio marketplace link. You can either install it directly from Visual Studio or download and install it from the Visual Studio marketplace.

[Visual Studio 2022](#)

[Visual Studio 2019 or lower](#)

Prerequisites

The following software prerequisites must be installed to install the Syncfusion ASP.NET MVC extension, as well as to creating, adding snippet, converting, and upgrading Syncfusion ASP.NET MVC applications.

- [Visual Studio 2013 or later](#).

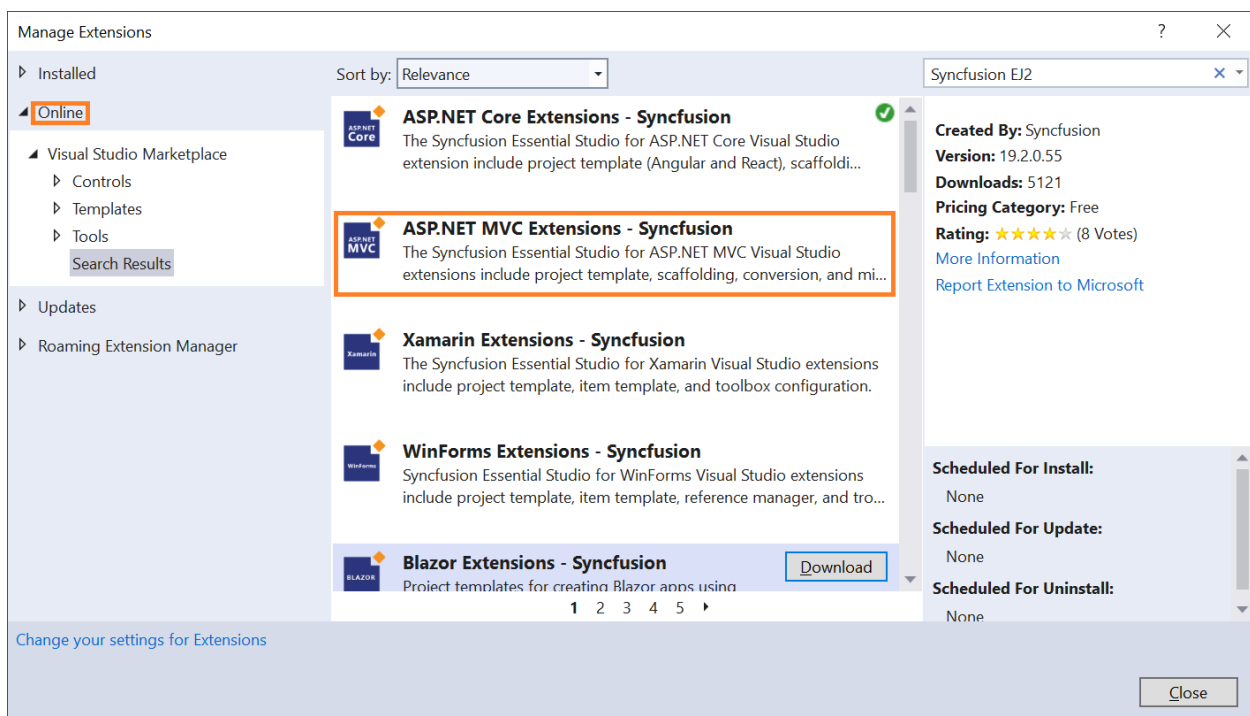
Install through the Visual Studio Manage Extensions

The steps below assist you to how to install the Syncfusion ASP.NET MVC extensions from **Visual Studio Manage Extensions**.

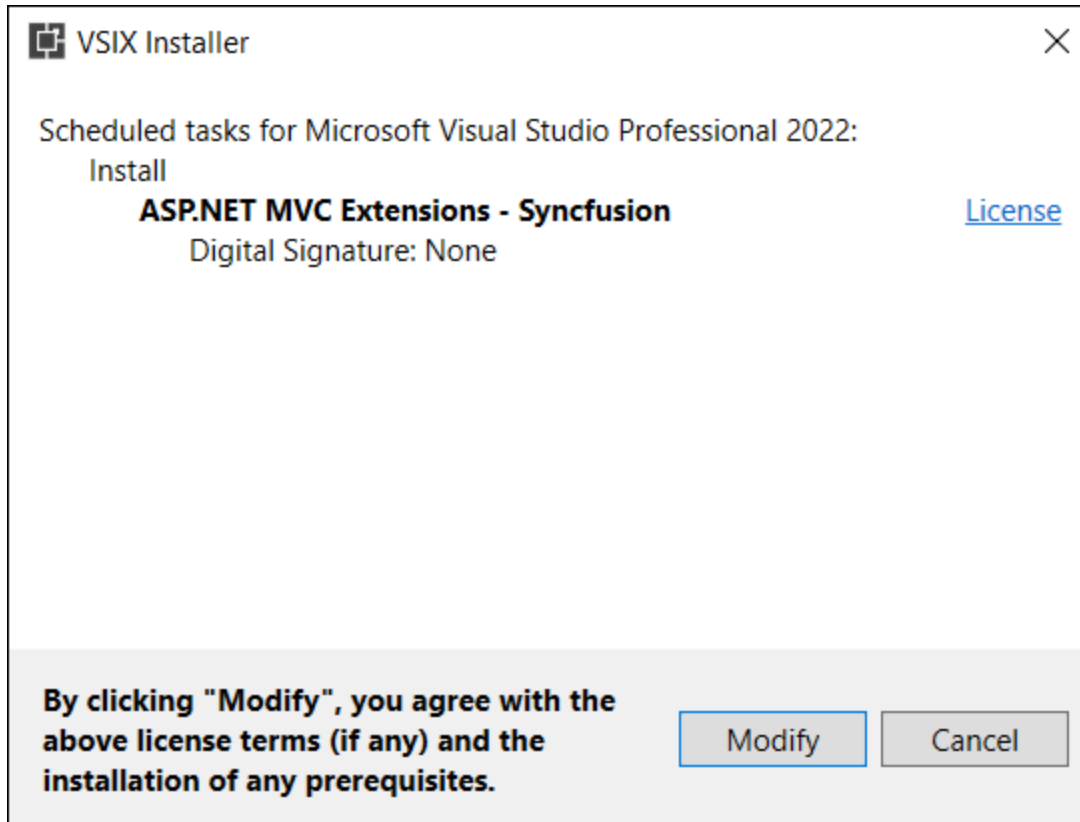
1. Open the Visual Studio.
2. Navigate to **Extension ->Manage Extensions** and open the Manage Extensions.

Note: In Visual Studio 2017 or lower, go to Tools -> Extensions and Updates.

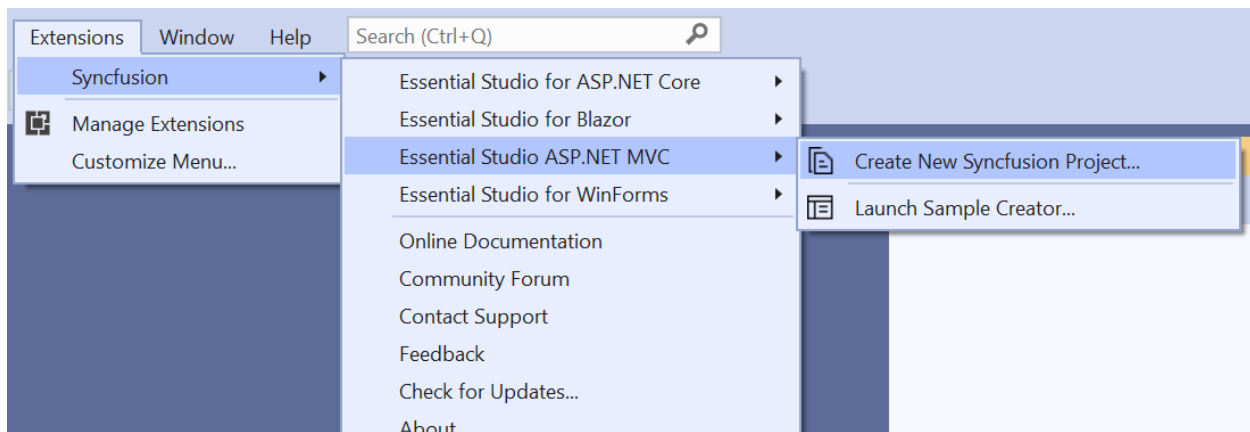
3. On the left, click the **Online** tab and type “**Syncfusion EJ2 MVC**” in the **search box**.



4. Click the **Download** button in the “**ASP.NET MVC Extension - Syncfusion**”.
5. Close all Visual Studio instances after downloading the extensions to begin the installation process. You will see the following VSIX installation prompt.



6. Click the **Modify** button.
7. After the installation is complete, open Visual Studio.
8. Now, under the menu **Extensions**, you can use the Syncfusion extensions from the Visual Studio.



Note: In Visual Studio 2017 or lower, you can see the Syncfusion menu directly in the Visual Studio menu.

[Install from the Visual Studio Marketplace](#)

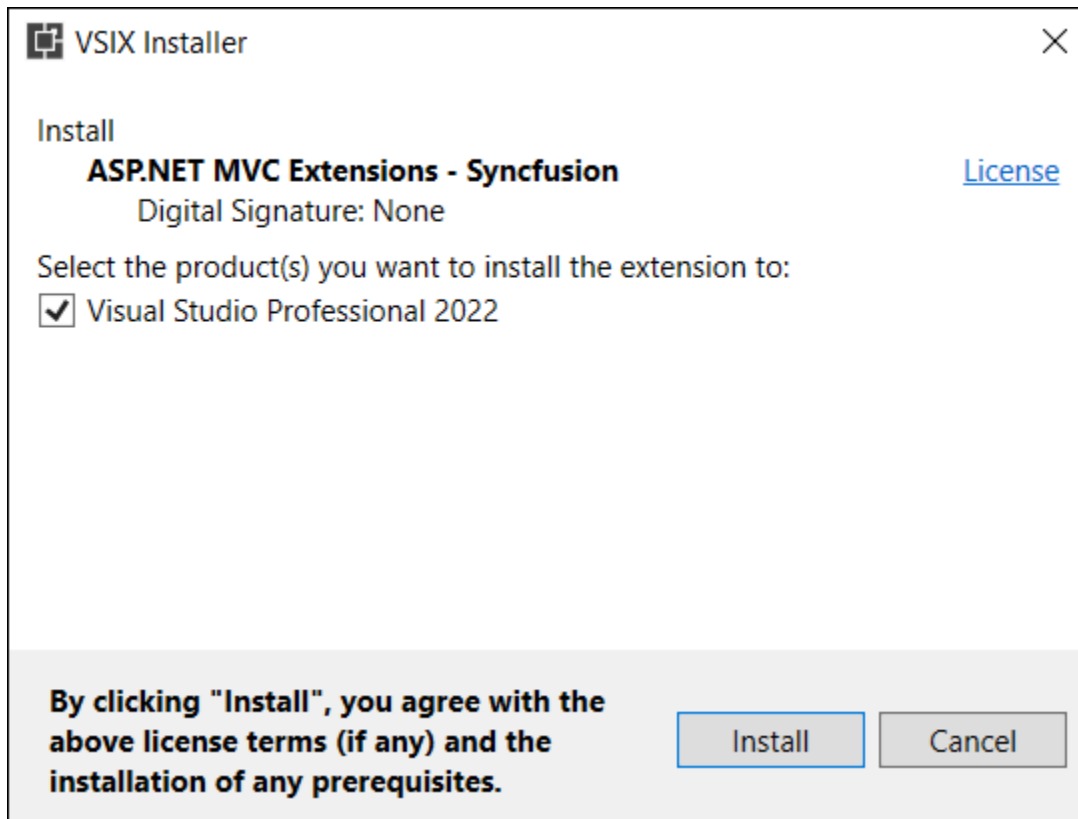
The steps below illustrate how to download and install the Syncfusion ASP.NET MVC extension from the Visual Studio Marketplace.

1. Download the Syncfusion ASP.NET MVC Extensions from the below Visual Studio Marketplace.

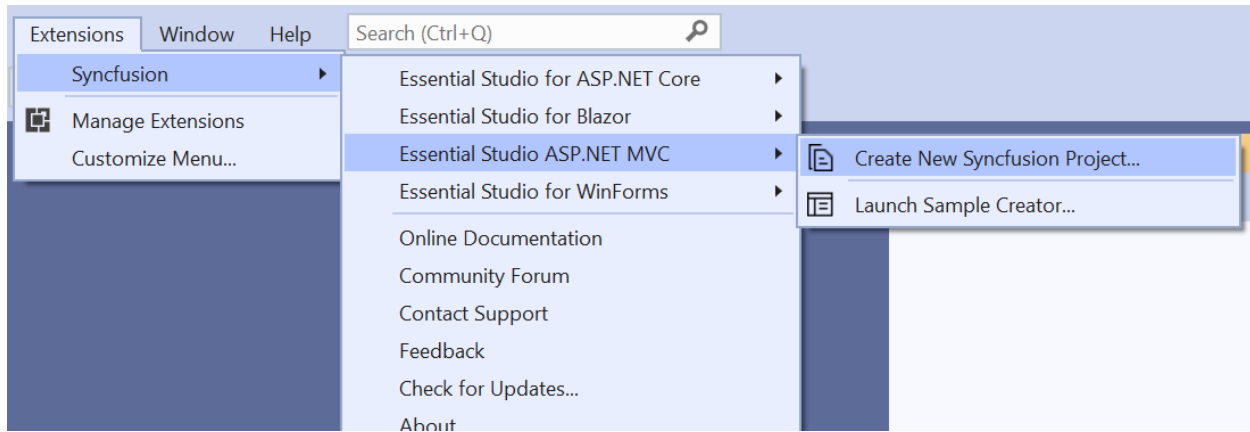
[Visual Studio 2022](#)

[Visual Studio 2019 or lower](#)

2. Close all Visual Studio instances running, if any.
3. Double-click to install the downloaded VSIX file. You will see the VSIX installation prompts with the corresponding installed Visual Studio version checkbox for select the Visual Studio to install extension.



4. Click the **Modify** button.
5. After the installation is complete, open Visual Studio 2019. You can now use Syncfusion extensions from the Visual Studio under the **Extensions** menu.



Note: In Visual Studio 2017 or lower, you can see the Syncfusion menu directly in the Visual Studio menu.

Visual Studio extensions

Create project

Syncfusion provides the **Visual Studio Project Templates** for the Syncfusion ASP.NET MVC platform to create the Syncfusion ASP.NET MVC Web Application with the Essential JS 2 components.

Note: The Syncfusion ASP.NET MVC (Essential JS 2) project templates are available from v16.2.0.41.

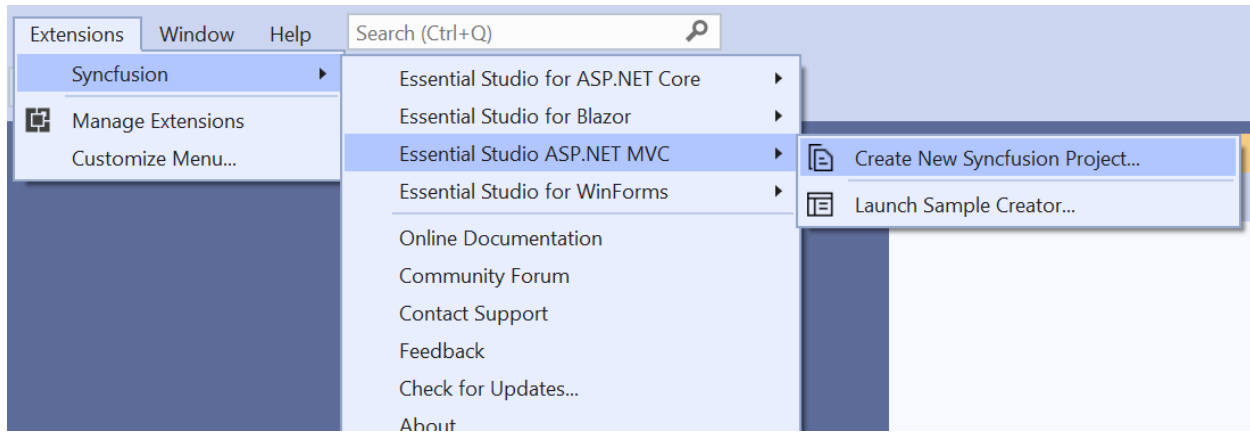
Use the following steps to create the **Syncfusion ASP.NET MVC (Essential JS 2) Web Application** through the **Visual Studio Project Template**.

Note: Before use the Syncfusion ASP.NET MVC Project Template, check whether the **ASP.NET MVC Extensions - Syncfusion** installed or not in Visual Studio Extension Manager by clicking on the **Extensions -> Manage Extensions -> Installed** for Visual Studio 2019 or later and for Visual Studio 2017 or lower by clicking on the **Tools -> Extensions and Updates -> Installed**. If this extension not installed, install the extension by follow the steps from the [download and installation](#) help topic.

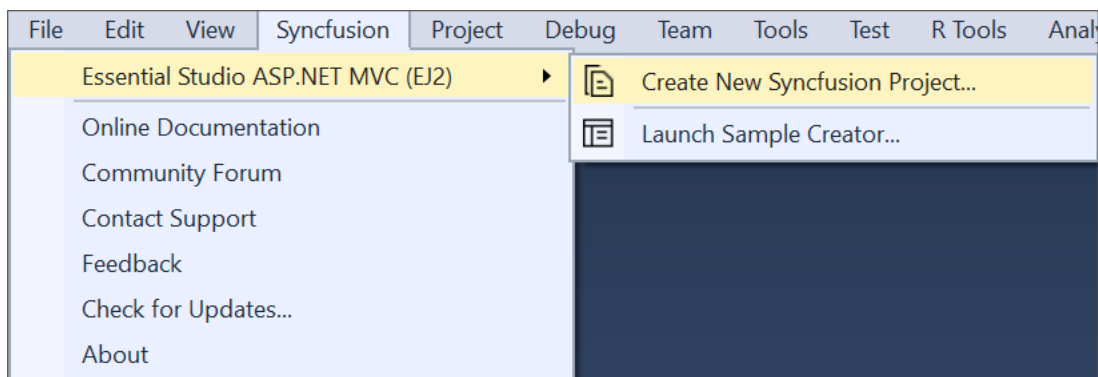
1. To create the Syncfusion ASP.NET MVC (Essential JS 2) project, follow either one of the options below:

Option 1:

Click **Extensions->Syncfusion** Menu and choose **Essential Studio for ASP.NET MVC (EJ2) > Create New Syncfusion Project...** in Visual Studio menu.

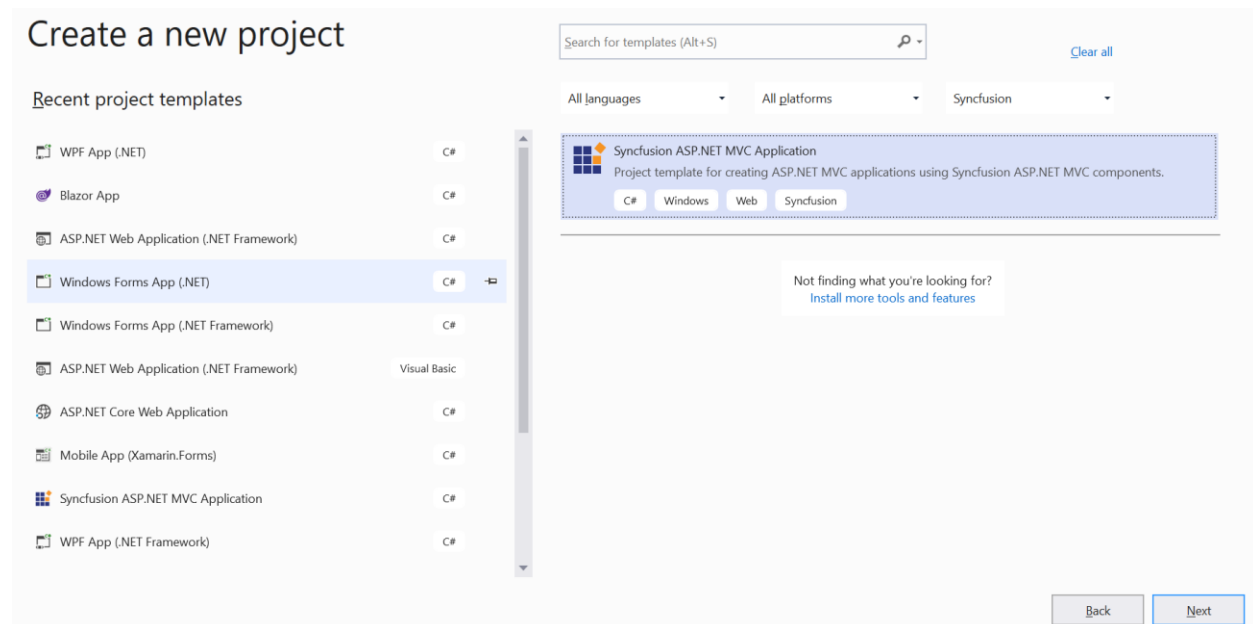


Note: In Visual Studio 2017 or lower, you can see the Syncfusion menu directly in the Visual Studio menu.

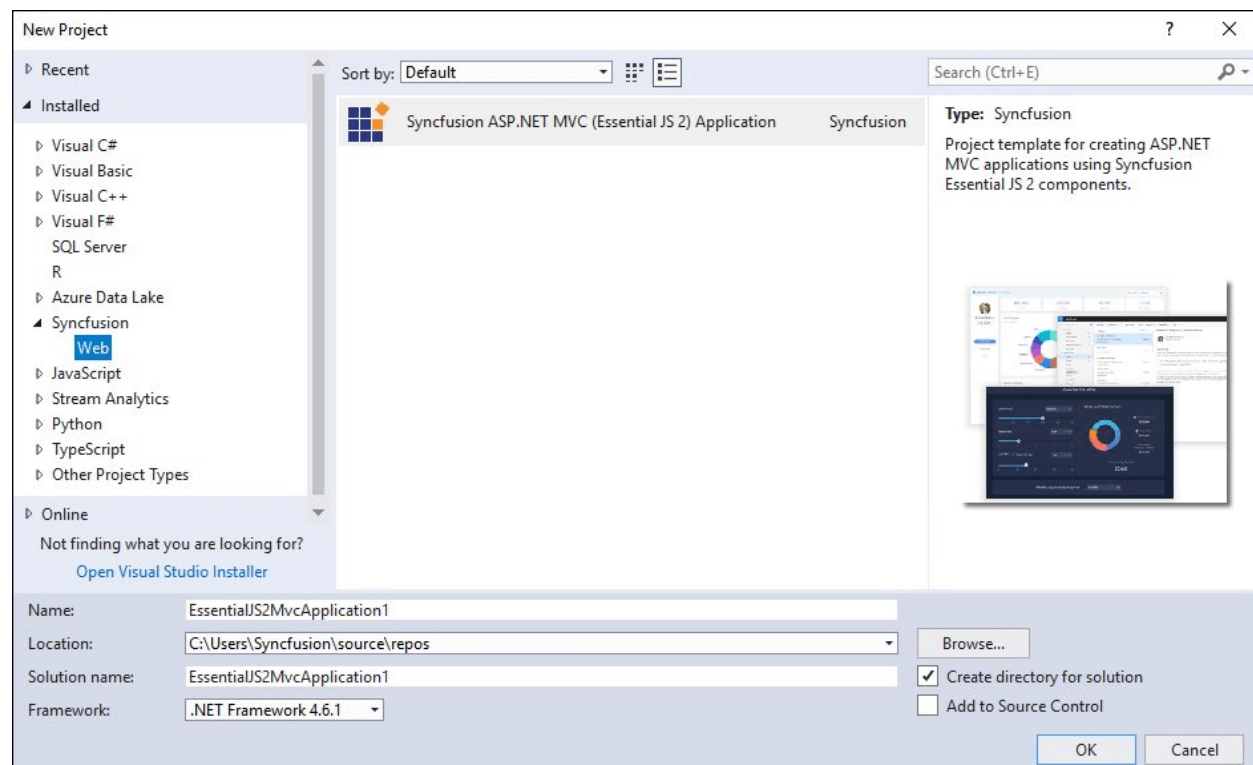


Option 2:

Choose **File -> New -> Project**. Opens a new dialog to create a new project. By filtering the project type with Syncfusion or using the **Syncfusion** keyword in the search option, you can get the templates offered by Syncfusion for ASP.NET MVC.



Note: In Visual Studio 2017 or lower, Choose **File > New > Project** and navigate to **Syncfusion > Web > Syncfusion ASP.NET MVC (Essential JS 2) Application**.



2. Name the **Project**, choose the **destination location**, and set the .NET Framework of the project, and then click **OK**. The Project Configuration Wizard appears.

ASP.NET MVC project configuration wizard

Version 19.4.0.38

×

Project Preference

Target MVC Version

MVC5

Themes

Bootstrap v5

Assets From

NuGet

Theme Preview

Preview for your selected theme is shown as follows:

Button

Primary

Grid

PRODUCT ID	PRODUCT NAME	UNIT PRICE	UNITS IN STOCK
1	Chai	\$18.00	39
2	Chang	\$19.00	17
3	Aniseed Syrup	\$10.00	13

«

<

1

2

3

4

5

...

>

»

1 to 5 of 77

Syncfusion

Help

Support

Suggest a Feature

What's New

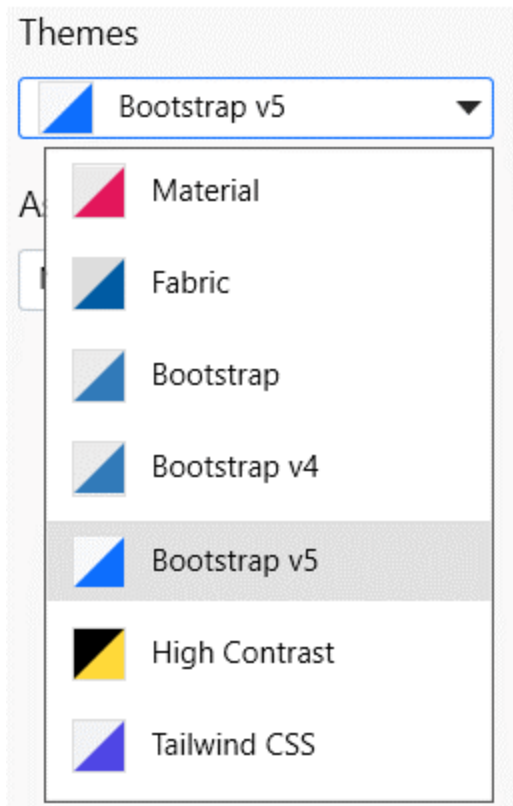
Create

Cancel

Project configurations

Target MVC Version: Select the version of ASP.NET MVC Project, either MVC5 or MVC4.

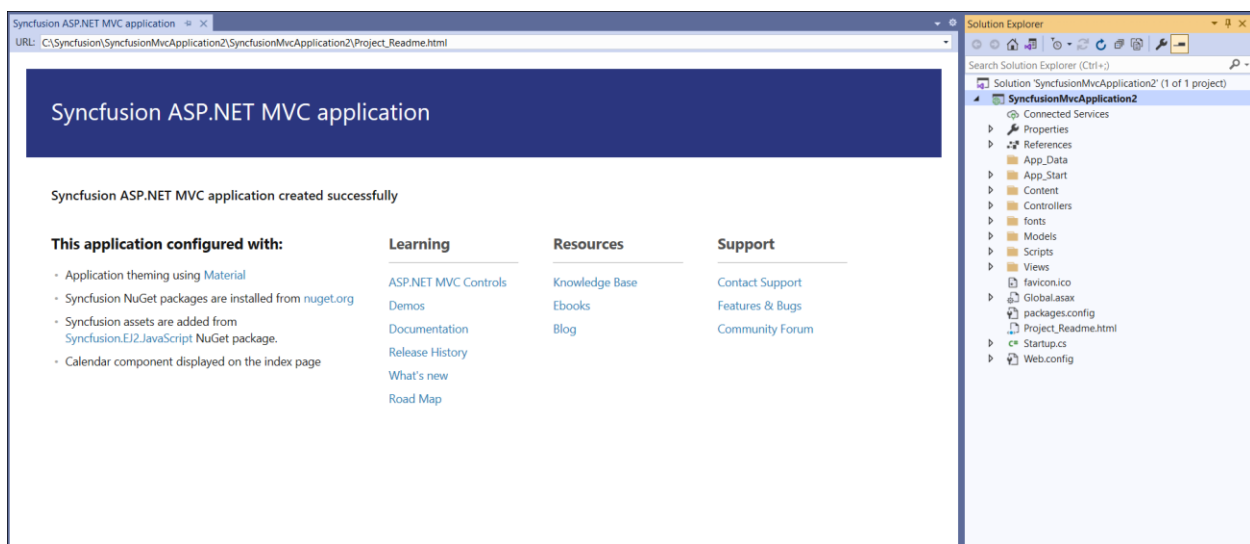
Theme Selection: Choose the required Theme. The Theme Preview section shows the controls preview with selected theme before creating the Syncfusion project.



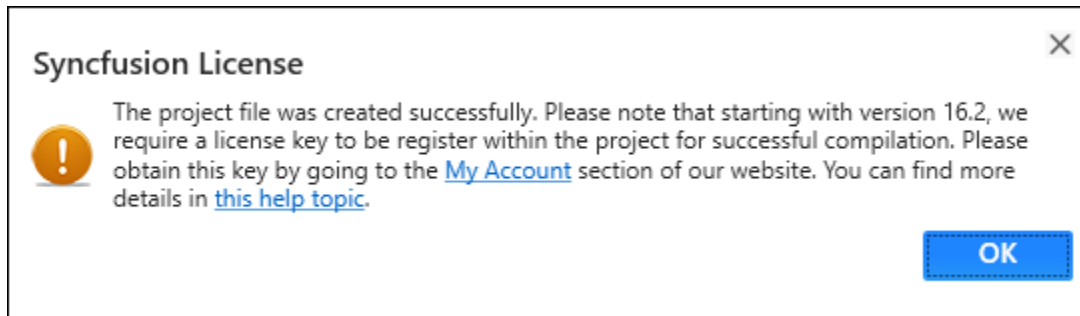
Assets From: : Load the Syncfusion Essential JS 2 assets to ASP.NET MVC Project, either NuGet, CDN, or Installed Location.

Note: Installed location option will be available only when the Syncfusion Essential JavaScript 2 setup has been installed.

3. Click **Create**, the Syncfusion ASP.NET MVC (Essential JS 2) Application will be created.



4. The created Syncfusion ASP.NET MVC application configures with most recent Syncfusion ASP.NET MVC NuGet packages, selected style and scripts for use Syncfusion components.
5. Then, the Syncfusion licensing registration required message box will be shown, if you installed the trial setup or NuGet packages since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Navigate to the [help topic](#), which is shown in the licensing message box to generate and register the Syncfusion license key to your project. Refer to this [blog](#) post to learn more about the licensing changes introduced in Essential Studio.



Converting ASP.NET MVC application to ASP.NET MVC application

Syncfusion ASP.NET MVC conversion is a Visual Studio add-in that converts an existing ASP.NET MVC application into a Syncfusion ASP.NET MVC (Essential JS 2) Web application by adding the required assemblies and resource files.

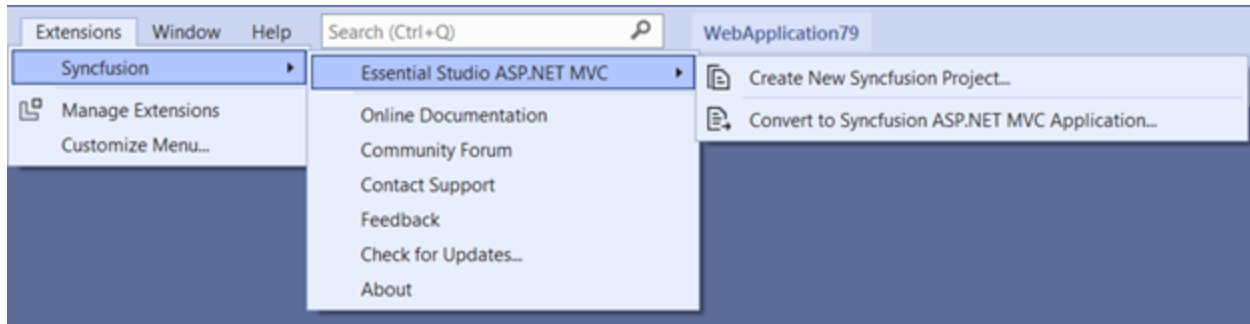
Note: The Syncfusion ASP.NET MVC (Essential JS 2) Web Application Project conversion utility is available from v16.3.0.17. Before use, the Syncfusion ASP.NET MVC Project Conversion, check whether the **ASP.NET MVC Extensions - Syncfusion** installed or not in Visual Studio Extension Manager by clicking on the **Extensions -> Manage Extensions -> Installed** for Visual Studio 2019 or later and for Visual Studio 2017 or lower by clicking on the **Tools -> Extensions and Updates -> Installed**. If this extension not installed, install the extension by follow the steps from the [download and installation](#) help topic. Also, check whether the corresponding Essential Studio version build installed or not. If the Essential Studio version is not same for both the Extension and build, then the Project Conversion will not be shown.

The steps below help you to convert the ASP.NET MVC application to the Syncfusion ASP.NET MVC application via the Visual Studio:

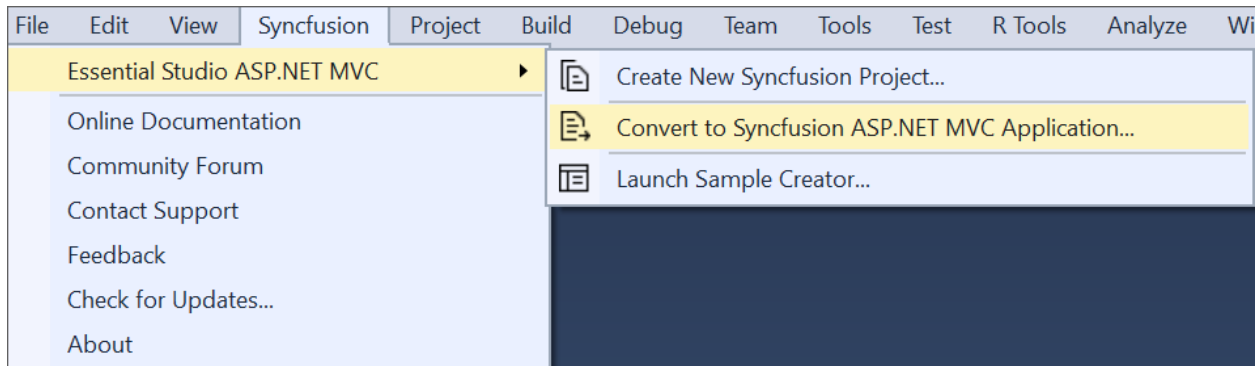
1. Open an existing Microsoft ASP.NET MVC Web Application or create a new Microsoft ASP.NET MVC Web Application.
2. To open the Syncfusion Project Conversion Wizard, follow either one of the options below:

Option 1:

Click **Extensions->Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC > Convert to Syncfusion ASP.NET MVC Application...** in **Visual Studio Menu**.

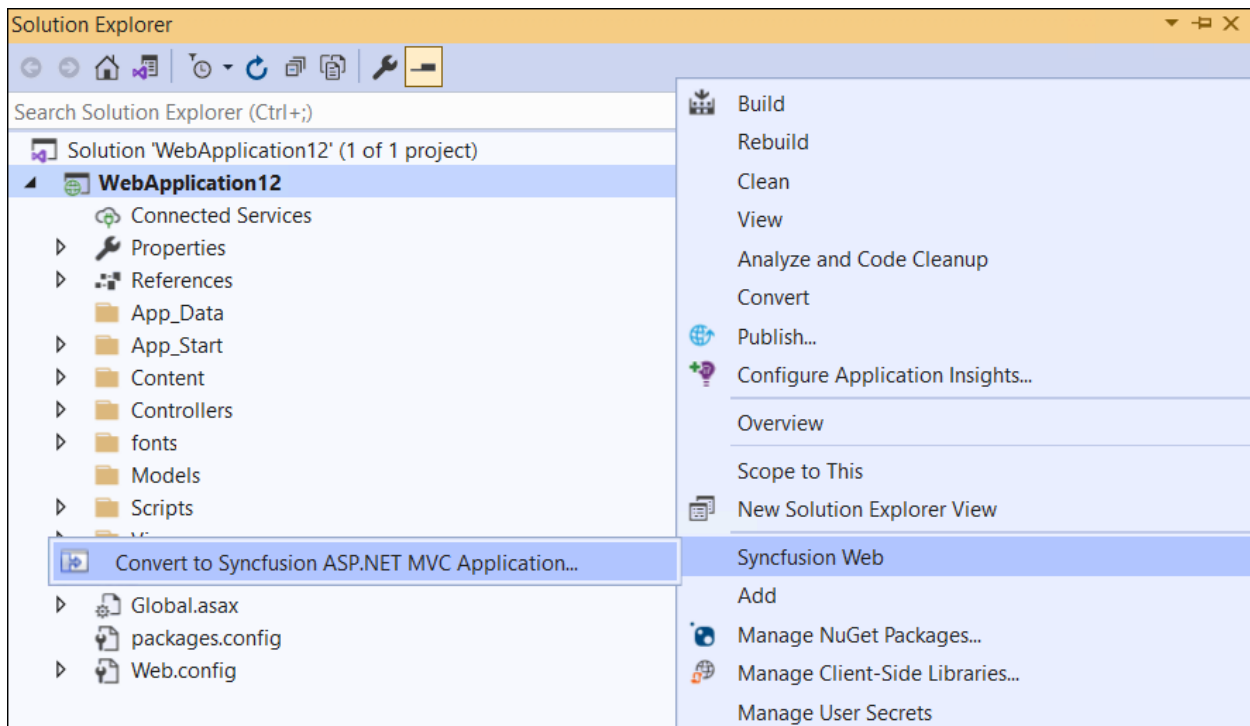


Note: In Visual Studio 2017 or lower, Click Syncfusion Menu and choose Essential Studio for ASP.NET MVC > Convert to Syncfusion ASP.NET MVC Application... in Visual Studio Menu.

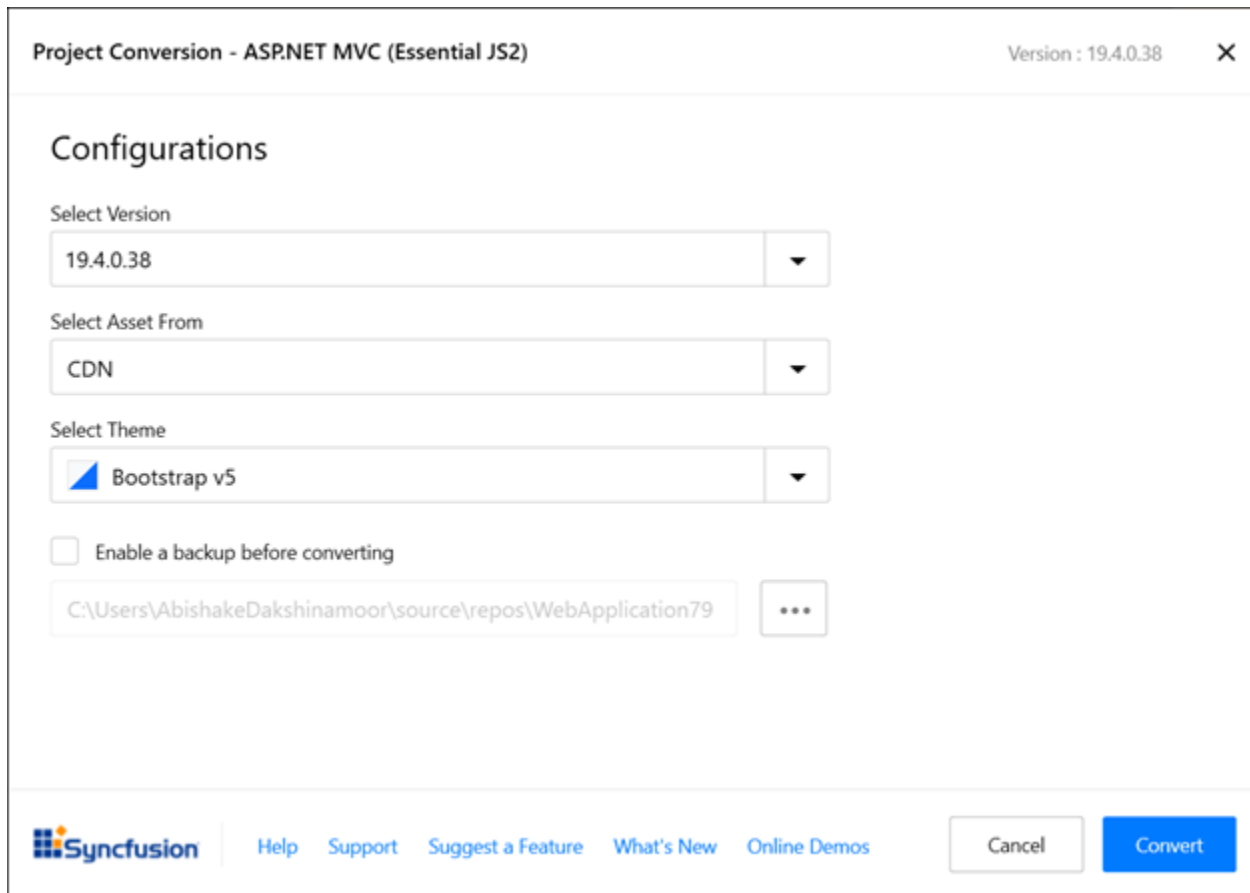


Option 2:

Right-click the **Project** from Solution Explorer, select **Syncfusion Web**, and choose the **Convert to Syncfusion ASP.NET MVC Application...** Refer to the following screenshot for more information.



3. The Syncfusion ASP.NET MVC Project Conversion window will appear. You can choose the required version of Syncfusion ASP.NET MVC, Assets from, and Themes to convert the application.



Project Conversion - ASP.NET MVC (Essential JS2) Version : 19.4.0.38

Configurations

Select Version
19.4.0.38

Select Asset From
CDN

Select Theme
Bootstrap v5

☐ Enable a backup before converting

C:\Users\AbishakeDakshinamoor\source\repos\WebApplication79

Syncfusion Help Support Suggest a Feature What's New Online Demos Cancel Convert

Note: The versions are loaded from the Syncfusion ASP.NET MVC NuGet packages which published in [NuGet.org](https://www.nuget.org) and it requires internet connectivity.

The following configurations are used in the Project conversion wizard.

Assets From: Load the Syncfusion Essential JS 2 assets to ASP.NET MVC Project, from either NuGet, CDN, or Installed Location.

Note: Installed location option will be available only when the Syncfusion Essential JavaScript 2 setup has been installed.

Choose the Theme: Choose the required theme.

4. Check the **“Enable a backup before converting”** checkbox if you want to take the project backup and choose the location.
5. The required Syncfusion ASP.NET MVC NuGet packages with selected version, scripts and styles are included in the ASP.NET MVC Web Application.

if you enabled project backup before converting, the old project was saved in the specified backup path location, as shown below once the conversion process completed.

> This PC > Local Disk (C:) > Users > Syncfusion > Source > Repos > WebApplication12 > WebApplication12_Backup_6.22.2021_9.06.36 >				
Name	Date modified	Type	Size	
App_Data	6/22/2021 9:06 AM	File folder		
App_Start	6/22/2021 9:06 AM	File folder		
bin	6/22/2021 9:06 AM	File folder		
Content	6/22/2021 9:06 AM	File folder		
Controllers	6/22/2021 9:06 AM	File folder		
fonts	6/22/2021 9:06 AM	File folder		
Models	6/22/2021 9:06 AM	File folder		
obj	6/22/2021 9:06 AM	File folder		
Properties	6/22/2021 9:06 AM	File folder		
Scripts	6/22/2021 9:06 AM	File folder		
Views	6/22/2021 9:06 AM	File folder		
favicon	6/21/2021 6:13 PM	Icon	32 KB	
Global.asax	6/21/2021 6:13 PM	ASP.NET Server Ap...	1 KB	
Global.asax.cs	6/21/2021 6:13 PM	C# Source File	1 KB	
packages.config	6/21/2021 6:17 PM	XML Configuration...	2 KB	
Web.config	6/21/2021 6:17 PM	XML Configuration...	4 KB	
Web.Debug.config	6/21/2021 6:13 PM	XML Configuration...	2 KB	
Web.Release.config	6/21/2021 6:13 PM	XML Configuration...	2 KB	
WebApplication12	6/21/2021 6:17 PM	Visual C# Project fi...	12 KB	
WebApplication12.csproj.user	6/21/2021 6:17 PM	Per-User Project O...	2 KB	

- If you installed the trial setup or NuGet packages from nuget.org you must register the Syncfusion license key to your project since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Navigate to the [help topic](#) to generate and register the Syncfusion license key to your project. Refer to this [blog](#) post for understanding the licensing changes introduced in Essential Studio.

Upgrading Syncfusion ASP.NET MVC application to latest version

The Syncfusion ASP.NET MVC migration add-in for Visual Studio allows you to migrate an existing Syncfusion ASP.NET MVC application from one version of Essential Studio version to another version. This reduces the amount of manual work required when migrating the Syncfusion version.

IMPORTANT

The Syncfusion ASP.NET MVC (Essential JS 2) Web Application Project Migration utility is available from v16.3.0.17.

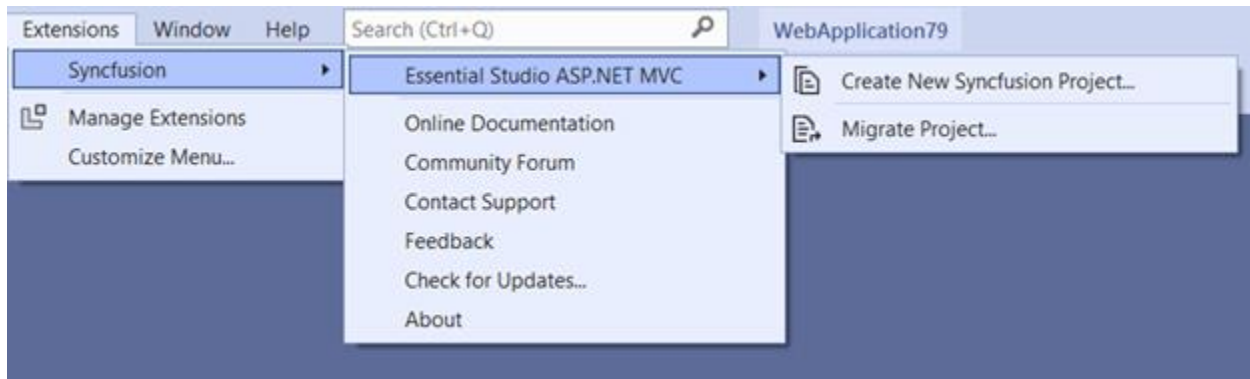
Note: Before use, the Syncfusion ASP.NET MVC Project Migration, check whether the **ASP.NET MVC Extensions - Syncfusion** installed or not in Visual Studio Extension Manager by clicking on the **Extensions -> Manage Extensions -> Installed** for Visual Studio 2019 or later and for Visual Studio 2017 or lower by clicking on the **Tools -> Extensions and Updates -> Installed**. If this extension not installed, install the extension by follow the steps from the [download and installation](#) help topic. Also, check whether the corresponding Essential Studio version build installed or not. If the Essential Studio version is not same for both the Extension and build, then the Project Migration will not be shown.

The steps below will assist you to upgrade the Syncfusion version in the Syncfusion ASP.NET MVC application via Visual Studio 2019:

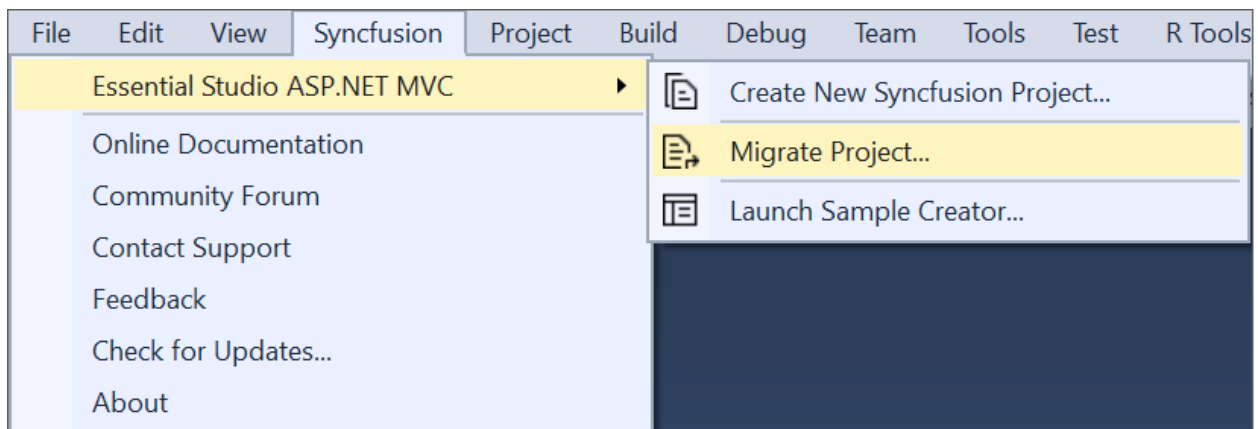
1. Open the Syncfusion ASP.NET MVC application that uses the Syncfusion component.
2. To open the Migration Wizard, either one of the following options should be followed:

Option 1:

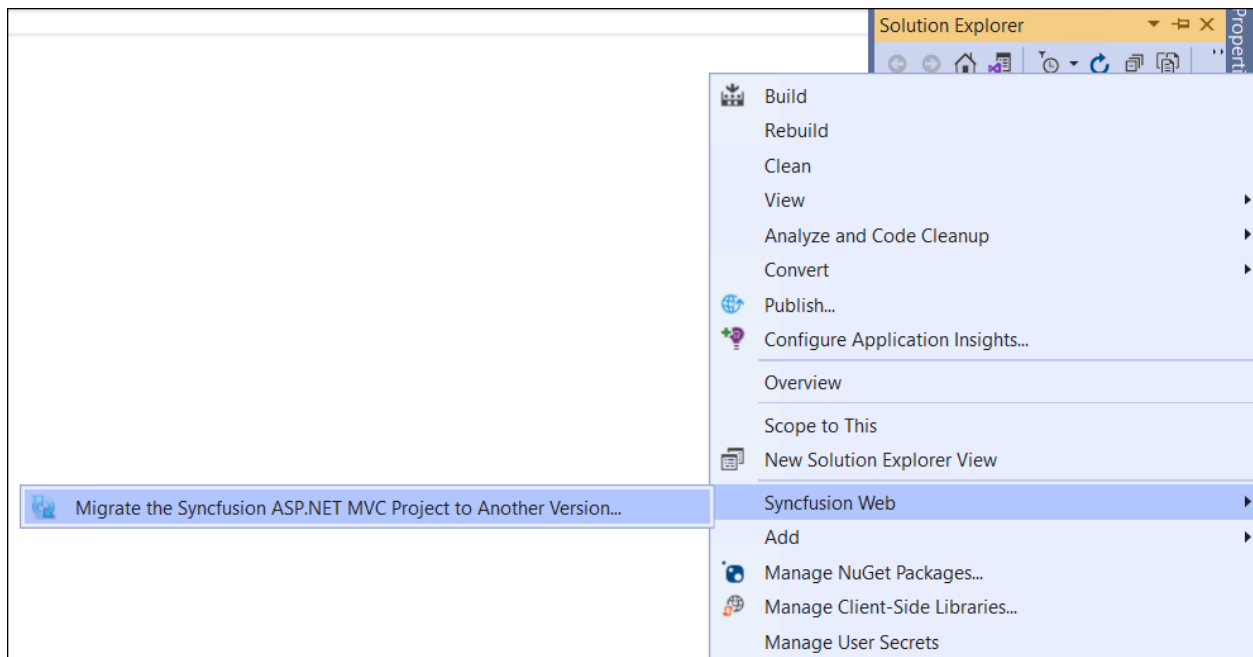
Click **Extensions->Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC > Migrate Project...** in **Visual Studio Menu**.



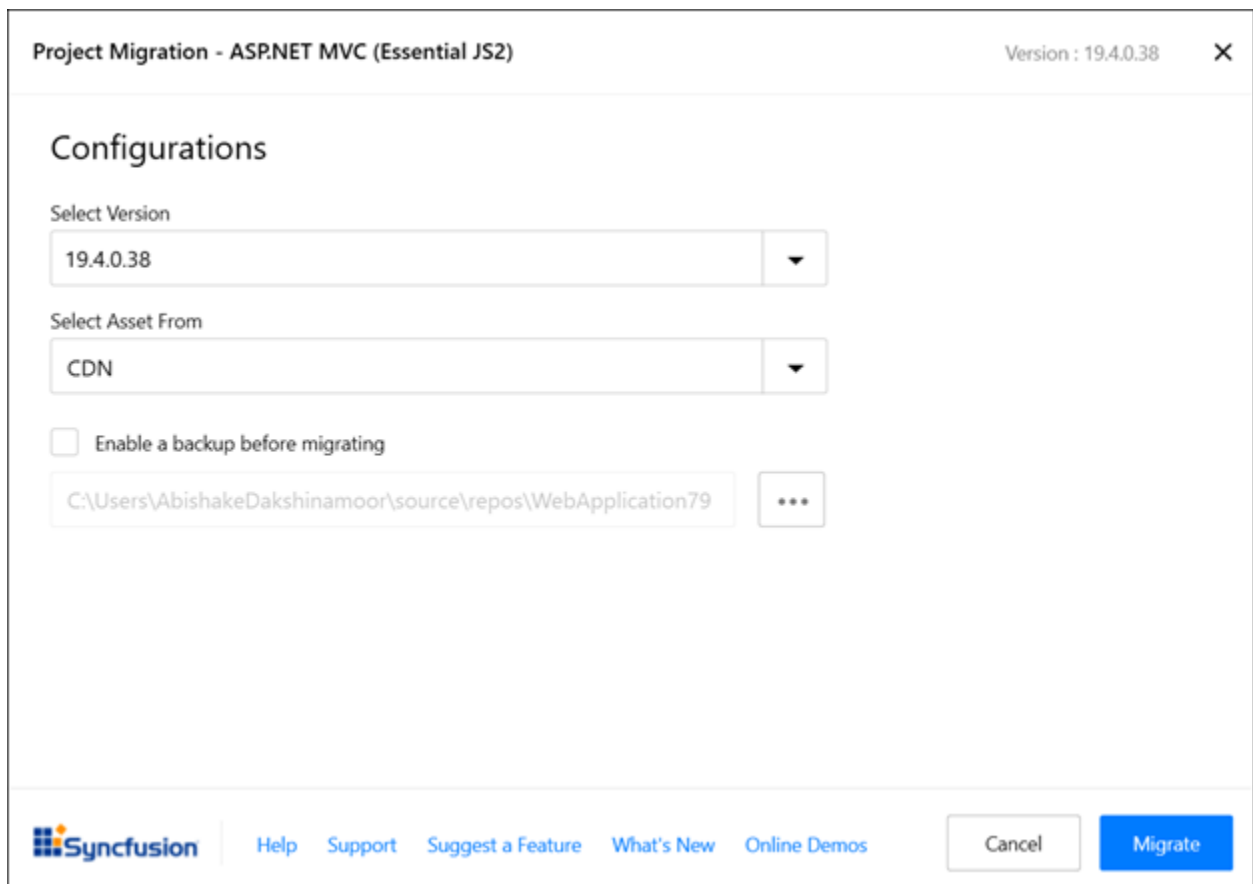
Note: In Visual Studio 2017 or lower, Click Syncfusion Menu and choose Essential Studio for ASP.NET MVC > Migrate Project... in Visual Studio Menu.

**Option 2:**

Right-click the **Syncfusion ASP.NET MVC Application** from Solution Explorer and select **Syncfusion Web**. Choose **Migrate the Syncfusion ASP.NET MVC Project to Another Version...**



3. The Syncfusion Project Migration window will appear. You can choose the required version of Syncfusion ASP.NET MVC to migrate.



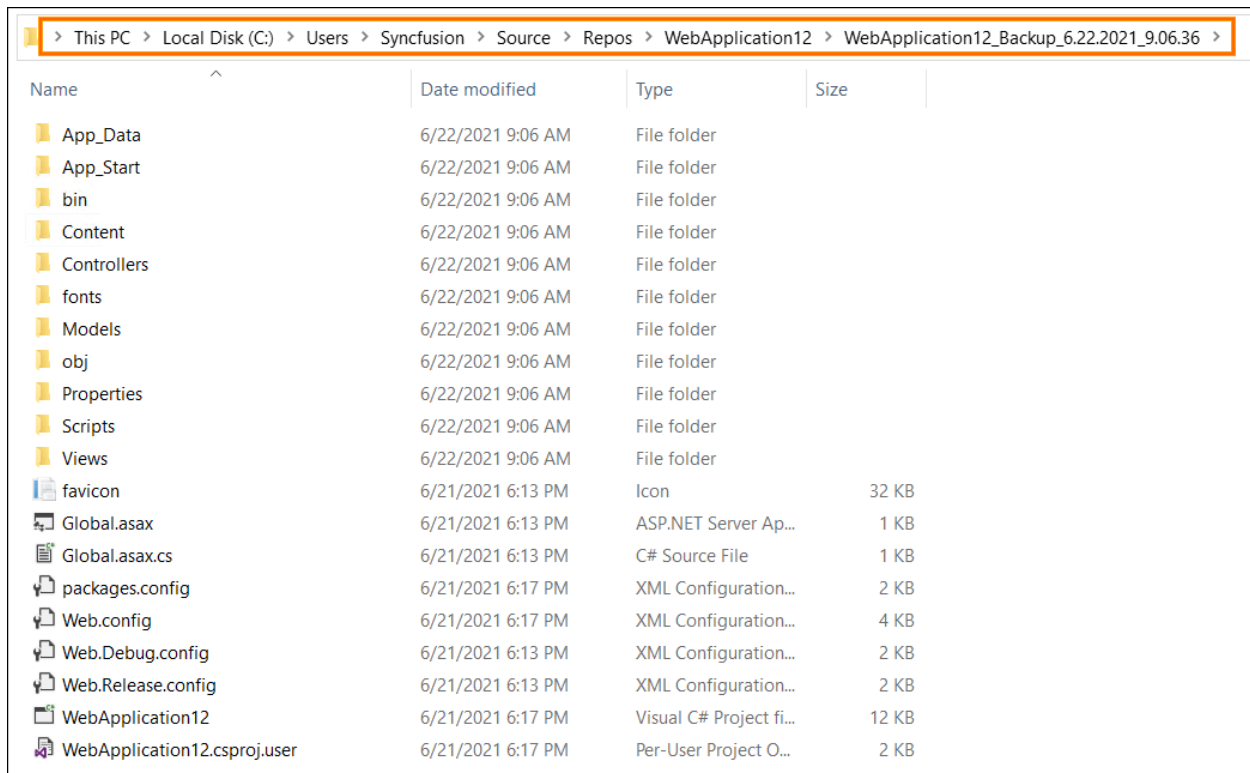
Note: The versions are loaded from the Syncfusion ASP.NET MVC NuGet packages which published in [NuGet.org](https://www.nuget.org) and it requires internet connectivity.

Assets From: Load the Syncfusion Essential JS 2 assets to ASP.NET MVC Project, from either NuGet, CDN or Installed Location.

Note: Installed location option will be available only when the Syncfusion Essential JavaScript 2 setup has been installed.

4. Check the “**Enable a backup before migrating**” checkbox if you want to take the project backup and choose location.
5. The Syncfusion Reference Assemblies, Scripts, and CSS are updated to the selected version in the project.

if you enabled project backup before migrating, the old project was saved in the specified backup path location, as shown below once the migration process completed



6. If you installed the trial setup or NuGet packages from [nuget.org](https://www.nuget.org) you must register the Syncfusion license key to your project since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Navigate to the [help topic](#) to generate and register the Syncfusion license key to your project. Refer to this [blog](#) post for understanding the licensing changes introduced in Essential Studio.

Create Sample in ASP.NET MVC Application

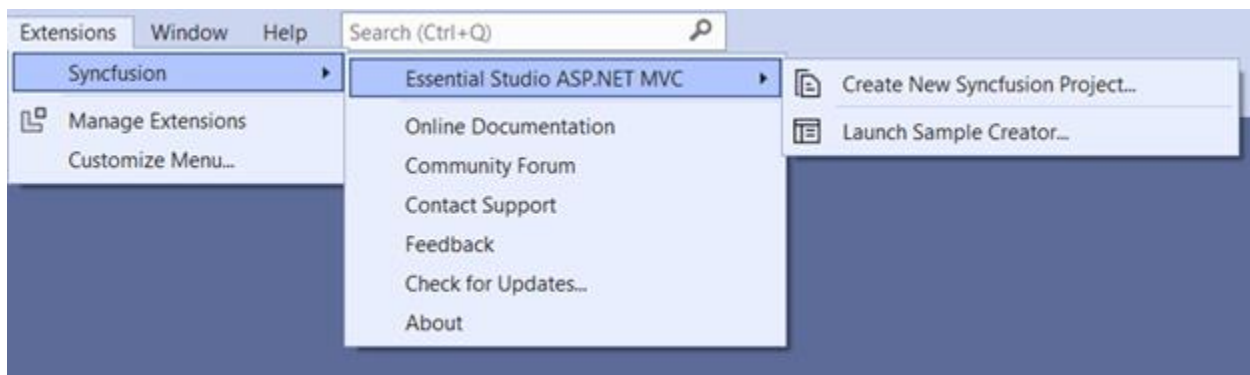
The Syncfusion Sample Creator is a tool that lets you make Syncfusion ASP.NET MVC (Essential JS 2) projects with sample code for required Syncfusion component features and Syncfusion control configuration.

Note: The Syncfusion ASP.NET MVC (Essential JS 2) Sample Creator utility is available from v16.3.0.17.

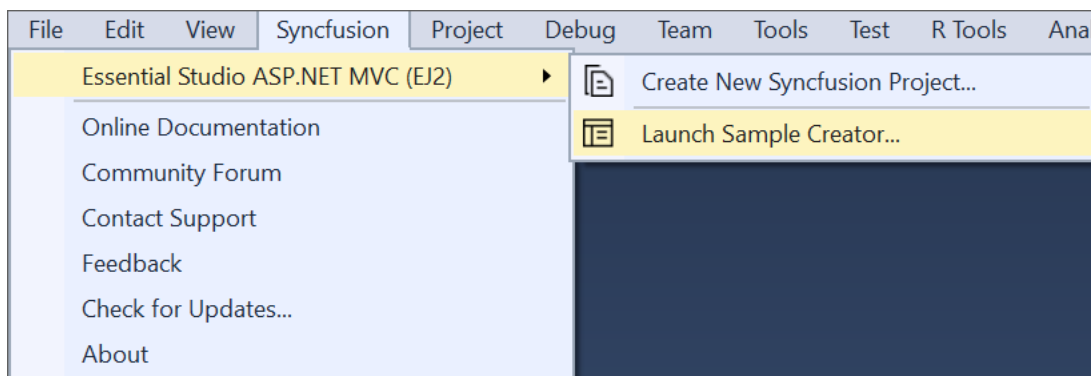
Use the following steps to create the Syncfusion ASP.NET MVC (Essential JS 2) Application through the Sample Creator utility:

1. Follow one of the options below to launch the ASP.NET MVC (Essential JS 2) Sample Creator application:

Option 1: Click **Extensions->Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ2) > Launch Sample Creator...** in Visual Studio.



Note: In Visual Studio 2017 or lower, Click Syncfusion Menu and choose Essential Studio for ASP.NET MVC (EJ2) > Launch Sample Creator... in Visual Studio.



Option 2:

Launch the Syncfusion ASP.NET MVC Control Panel. Select the Sample Creator button to launch the ASP.NET MVC Sample Creator application. Refer to the following screenshot for more information.

Syncfusion® | Essential Studio

ASP.NET MVC | Add On & Utilities | Documentation | Other Products

Version 19.4.0.38

ASP.NET MVC

Lightweight, modular, and touch-friendly ASP.NET MVC components

[Run Online Demos](#) [Sample Creator](#)

Resource

[What's New](#) | [Release Notes](#) | [Read Me](#) | [User Guide](#) | [API Reference](#)

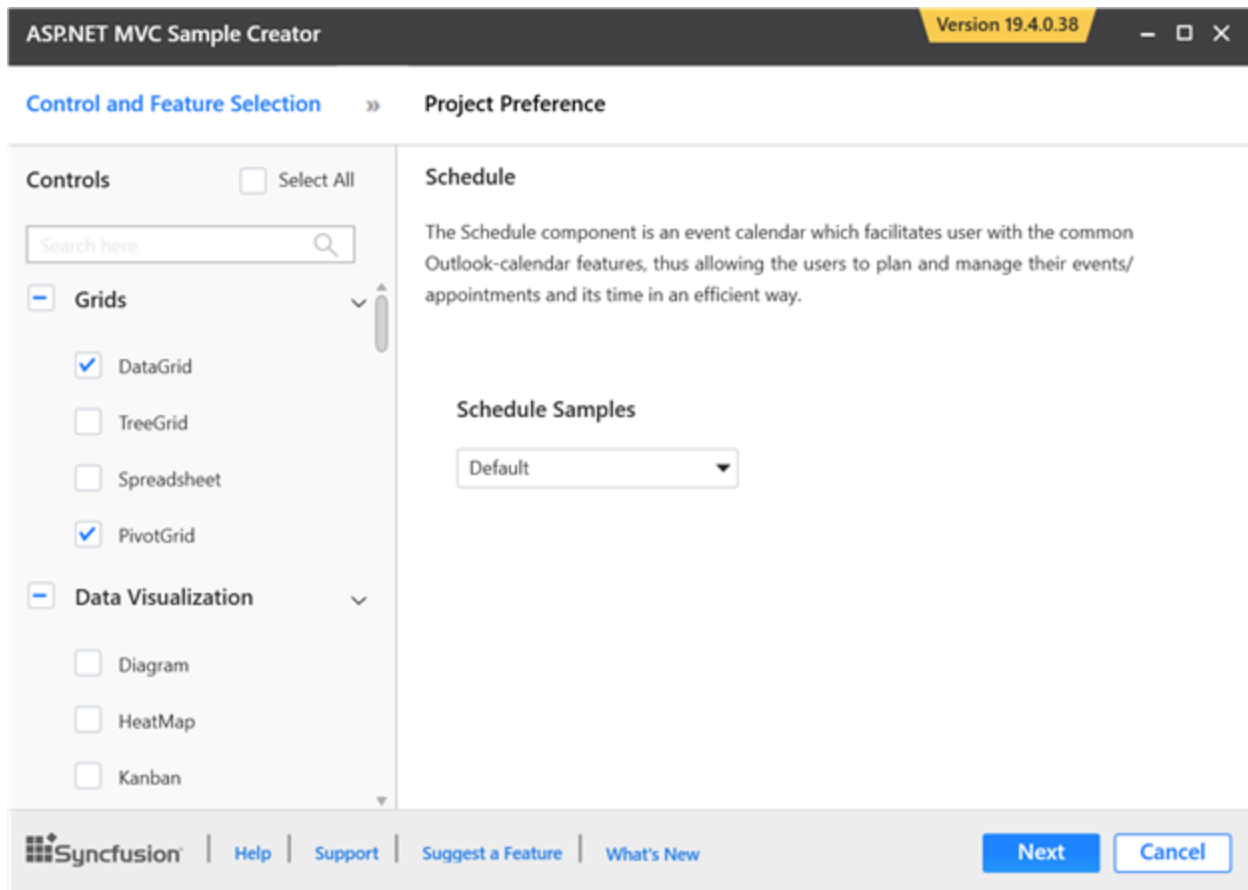
Licensed

[Sales](#) | [Support](#) | [Suggest a Feature](#) | [License Agreement](#) | [Check for Updates](#)

[View Installation Log](#) | [Recheck](#)

The screenshot displays the Syncfusion ASP.NET MVC website. The 'Sample Creator' button is highlighted with a red box. The page features the Syncfusion logo, navigation links for ASP.NET MVC, Add On & Utilities, Documentation, and Other Products. The version number 19.4.0.38 is shown in the top right. The main heading is 'ASP.NET MVC' with the tagline 'Lightweight, modular, and touch-friendly ASP.NET MVC components'. Below this are buttons for 'Run Online Demos' and 'Sample Creator'. A 'Resource' section includes links for 'What's New', 'Release Notes', 'Read Me', 'User Guide', and 'API Reference'. A 'Licensed' badge is also present. The bottom of the page contains links for 'Sales', 'Support', 'Suggest a Feature', 'License Agreement', 'Check for Updates', 'View Installation Log', and 'Recheck'. On the right side, there is a preview of the ASP.NET MVC application interface, showing a table with columns: 'Inventor Name', 'No of Patent Families', 'Active', and 'Main Fields of Invention'. The table lists inventors such as Kia Silvestrook, Paul Lapetun, Simon R. Walmsley, Tabon A. King, Akira Nakazawa, Garry R. Jackson, Norman M. Berry, Paul Ian Mackley, and Gregory McKinley. The table is filtered by 'Country Australia - 9 Items'. The interface also shows a sidebar with a search bar and a list of categories. The bottom right of the preview shows '1 of 12 pages (128 h)'.

2. Syncfusion controls and features are listed in the ASP.NET MVC Sample Creator.



Controls Selection: Choose the required controls. The controls are grouped with Syncfusion products.

Controls ☐ Select All

Search here

☒ **Grids**

- ☒ DataGrid
- ☒ PivotGrid

☐ **Data Visualization**

- ☐ Diagram
- ☐ HeatMap
- ☒ Chart
- ☒ AccumulationChart

Feature Selection: Based on the controls, the feature is enabled to choose the features of the corresponding controls.

DataGrid

The Grid component is used to display and manipulate tabular data with configuration options to control the way the data is presented. It can pull data from data sources such as an array of JavaScript objects, OData web services, or DataManager and binding data fields to columns. This sample demonstrates the most key features of the grid component such as paging, sorting, filtering, searching, and grouping, etc.

- ☒ Sorting
- ☒ Filtering
- ☒ Aggregates
- ☒ Column Resizing
- ☒ Grouping
- ☒ Selection
- ☒ Paging

Grid Samples **Editing**

Default Normal

Project Configuration

1. You can configure the project with the following details:
 - **MVC Version:** Choose the required MVC Version.
 - **VS Version:** Choose the Visual Studio version and Framework.
 - **View Engine:** By default, Syncfusion supports only Razor view engine for ASP.NET MVC projects.
 - **Assets From:** Choose the Syncfusion Essential JS 2 assets to ASP.NET MVC Project, either NuGet, CDN or Installed Location.

Note: Installed location option will be available only when the Syncfusion Essential JavaScript 2 setup has been installed.

- **Name:** Name your Syncfusion ASP.NET MVC Application.
- **Location:** Choose the target location of your project.
- **Theme Selection:** Choose the required theme. This section shows the controls preview before creating the Syncfusion project.

The screenshot shows the 'ASP.NET MVC Sample Creator' application window, version 19.4.0.38. The 'Project Preference' tab is active, displaying configuration options for the project. The 'Control and Feature Selection' tab is also visible.

Control and Feature Selection

- MVC Version: MVC5
- VS Version: 2022, NET4.7.2
- View Engine: Razor
- Assets From: NuGet
- Name: SyncfusionMvcApplication2
- Location: C:\Syncfusion (with a Browse button)
- Theme Selection: Bootstrap v5

Theme Preview

Preview for your selected theme shown below.

Button: Primary

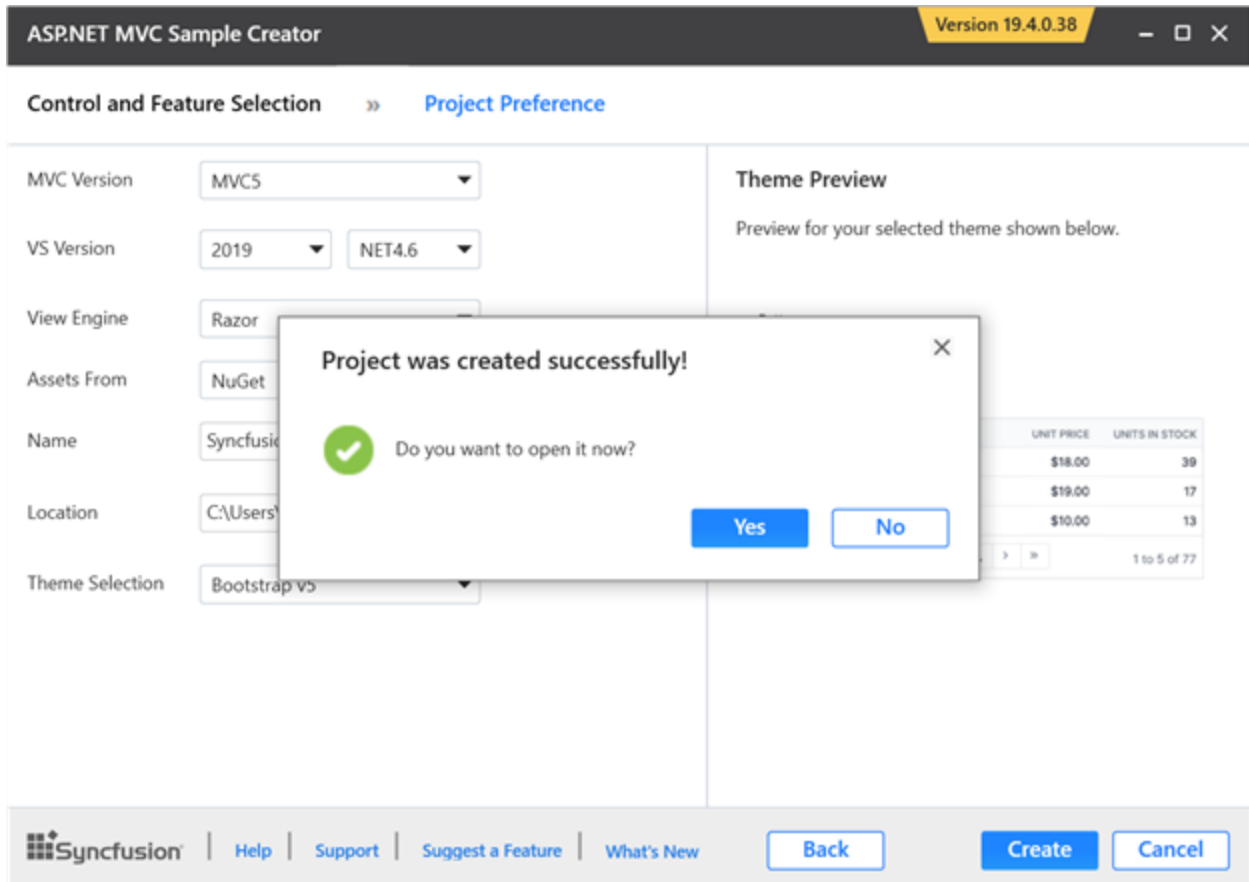
Grid:

PRODUCT ID	PRODUCT NAME	UNIT PRICE	UNITS IN STOCK
1	Chai	\$18.00	39
2	Chang	\$19.00	17
3	Aniseed Syrup	\$10.00	13

Navigation: 1 to 5 of 77

Footer: Syncfusion | Help | Support | Suggest a Feature | What's New | Back | Create | Cancel

2. Click **Create** button. After creating the project, open the project by clicking **Yes**. If you click **No**, the corresponding location of the project will be opened. Refer to the following screenshot for more information.



3. The new Syncfusion ASP.NET MVC sample is created. Required controller and view file for selected controls, selected style, scripts and corresponding Syncfusion NuGet packages are added in the created sample.

Scaffolding in ASP.NET MVC Application

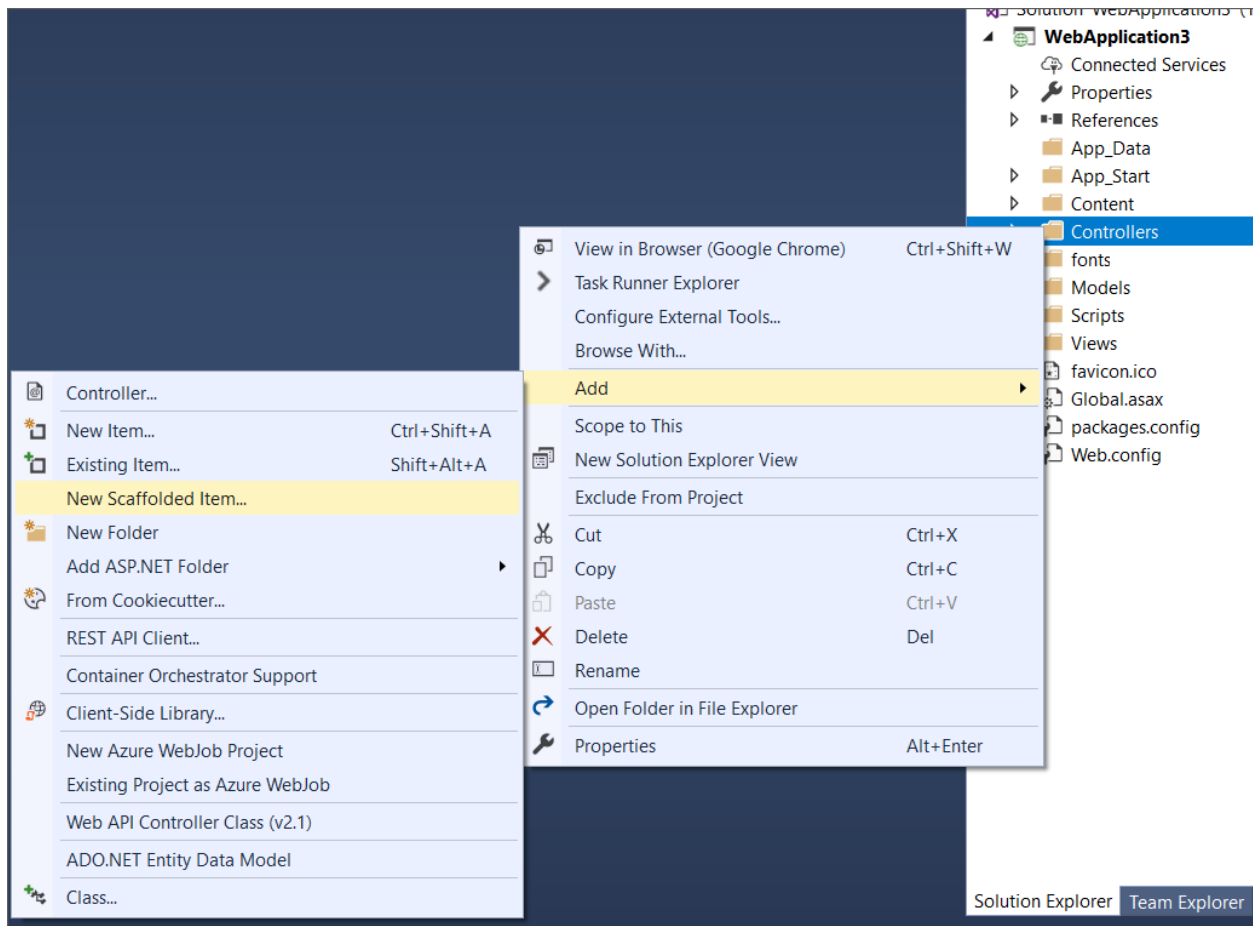
Syncfusion provides **Visual Studio Scaffolding** for Syncfusion ASP.NET MVC platform to quickly add code that interacts with data models and reduce the amount of time to develop with data operation in your project. Scaffolding provides an easier way to create Views and Controller action methods for Syncfusion ASP.NET MVC DataGrid, Charts, and Scheduler controls.

Note: The Syncfusion ASP.NET MVC UI Scaffolder is available from v16.4.0.40.

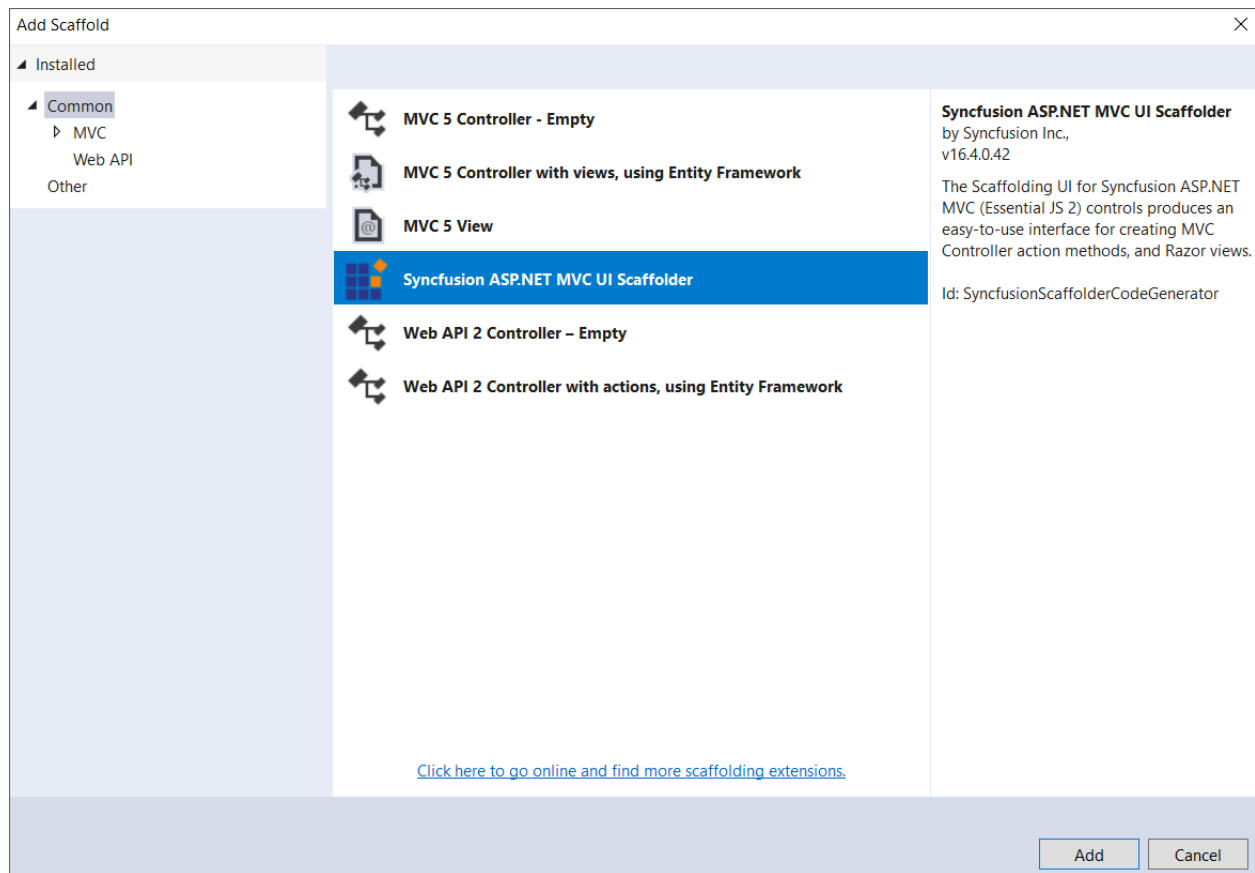
The following steps explain you how to add a scaffolded item to your ASP.NET MVC Web application.

Note: Before use, the Syncfusion ASP.NET MVC Scaffolding, check whether the **ASP.NET MVC Extensions - Syncfusion** installed or not in Visual Studio Extension Manager by clicking on the **Extensions** -> **Manage Extensions** -> **Installed** for Visual Studio 2019 or later and for Visual Studio 2017 or lower by clicking on the **Tools** -> **Extensions and Updates** -> **Installed**. If this extension not installed, install the extension by follow the steps from the [download and installation](#) help topic.

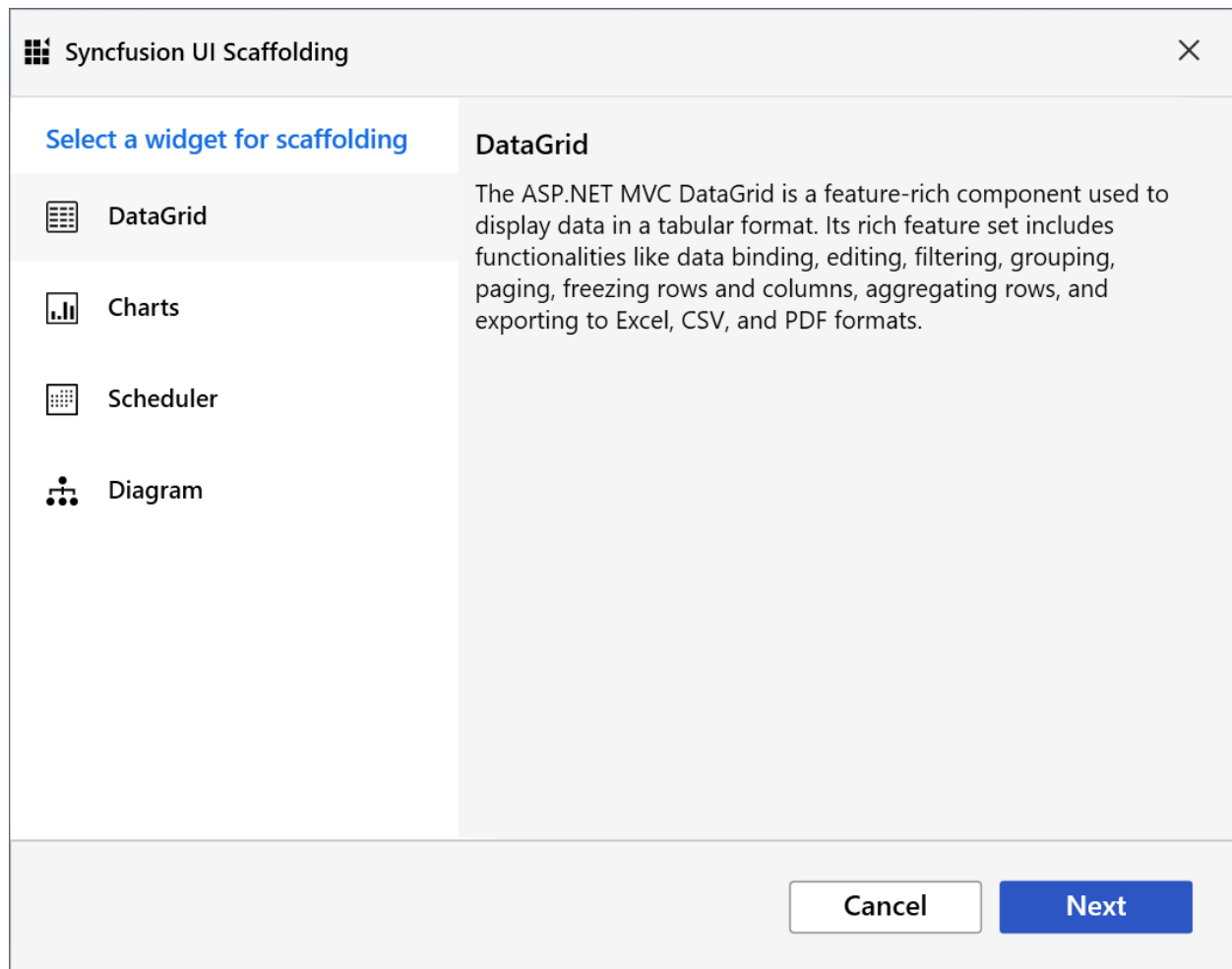
1. Right-click the **Controllers** folder in the Solution Explorer, click **Add**, and then select **New Scaffolded Item**.



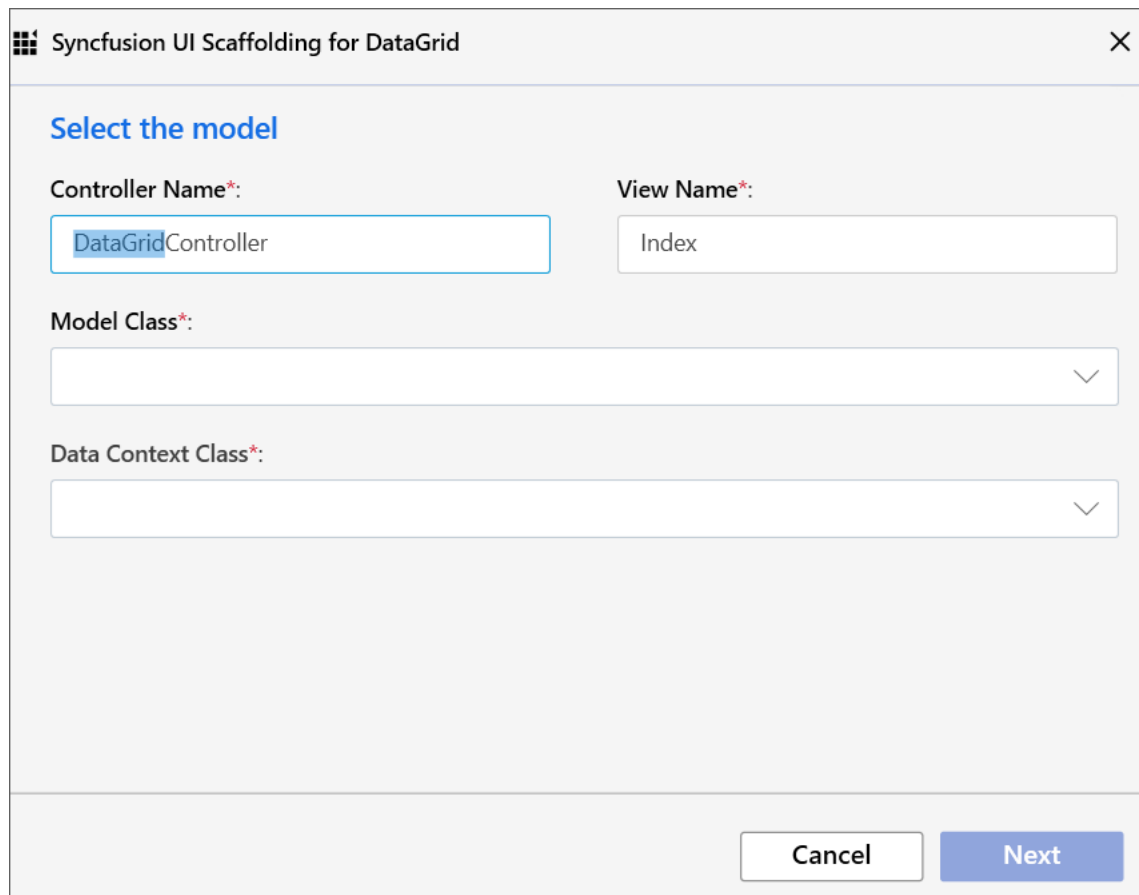
2. In the **Add Scaffold** dialog, select **Syncfusion ASP.NET MVC UI Scaffolder**, and then click **'Add'**.



3. In the Syncfusion UI Scaffolding dialog, select the desired control to perform scaffolding, and then click **Next**.



4. Selected control model dialogue will be launched in the Syncfusion UI Scaffolder. Enter the **Controller Name** and **View Name** as application requirements, and then select the required **Model Class** of the active project and its relevant **Data Context Class**, and then click **Next**.

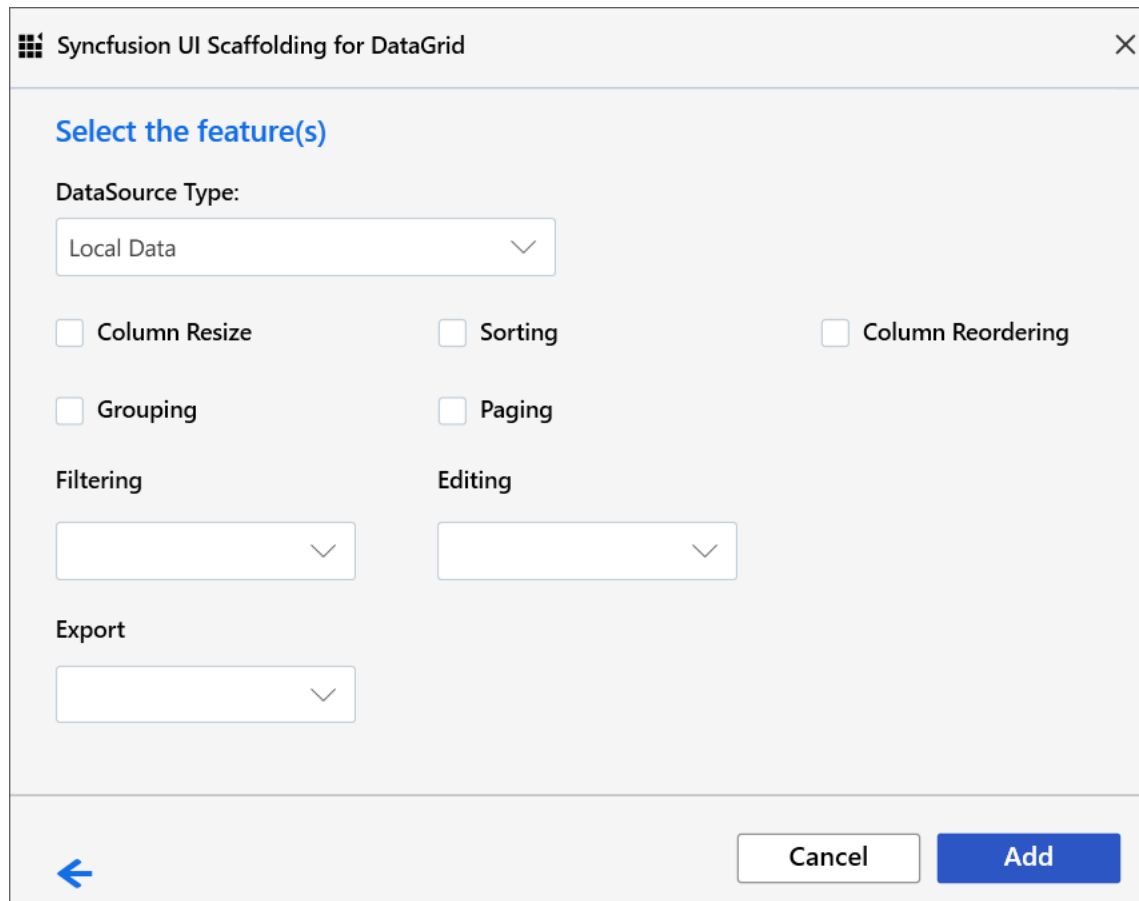


The image shows a dialog box titled "Syncfusion UI Scaffolding for DataGrid". It has a close button (X) in the top right corner. The main content area is titled "Select the model" in blue. Below this title, there are four input fields with labels and asterisks indicating they are required:

- Controller Name*:** A text box containing "DataGridController".
- View Name*:** A text box containing "Index".
- Model Class*:** A dropdown menu that is currently empty.
- Data Context Class*:** A dropdown menu that is currently empty.

At the bottom right of the dialog, there are two buttons: "Cancel" and "Next". The "Next" button is highlighted in blue.

5. Selected control feature dialogue will be launched in the Syncfusion UI Scaffolder. Select the required features, update the required data field, and then click **Add**.



Syncfusion UI Scaffolding for DataGrid

Select the feature(s)

DataSource Type:

Local Data

☐ Column Resize ☐ Sorting ☐ Column Reordering

☐ Grouping ☐ Paging

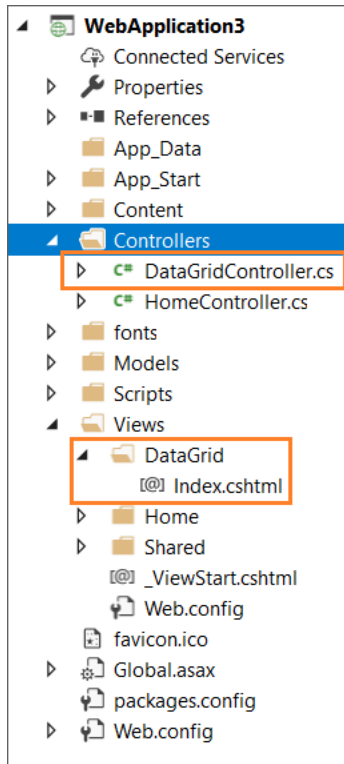
Filtering

Editing

Export

← Cancel Add

6. The **Controller** and the corresponding **View** files are now generated with the selected features of Syncfusion control code snippet.



7. If you installed the trial setup or NuGet packages from nuget.org you have to register the Syncfusion license key to your project since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Navigate to the [help topic](#) to generate and register the Syncfusion license key to your project. Refer to this [blog](#) post for understanding the licensing changes introduced in Essential Studio.

Note: Ensure that at least one Entity Framework model exists, and the application has been compiled once. If no Entity Framework model exist in your application, refer to this [documentation](#) to generate the Entity Framework model. After the model file has been added, ensure that the required DbContext and properties have been added. Now, build the application, and try scaffolding. If any changes have been done in the model properties, rebuild the application once before perform scaffolding.

Refer to the following UG links to render Syncfusion control after performed scaffolding.

MVC4: [Configure Essential JS 2 using Syncfusion.EJ2.MVC4 package](#)

MVC5: [Configure Essential JS 2 using Syncfusion.EJ2.MVC5 package](#)

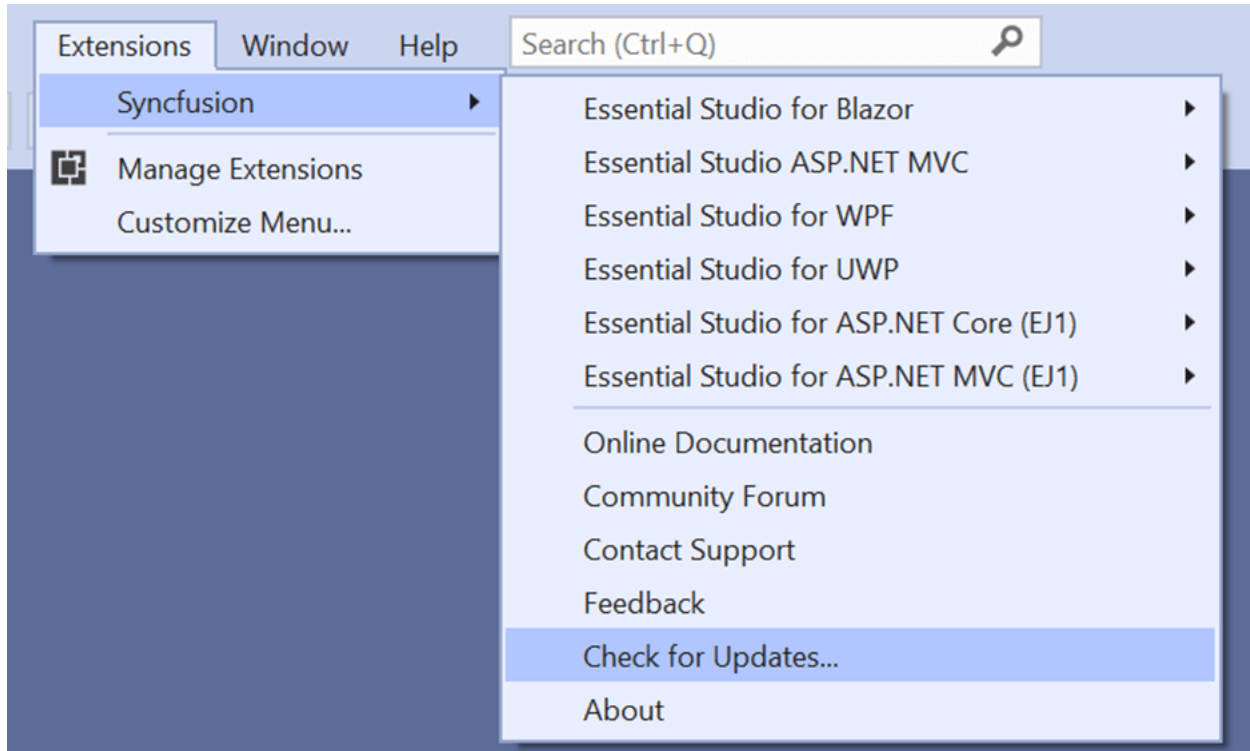
Check for Updates in ASP.NET MVC Application

Syncfusion provides the check for update extensions to find latest version of essential release was available, if it was available then provide option update most recent version of the Essential Studio release. So that, you always get the latest features, fixes, and improvements by installing the latest version.

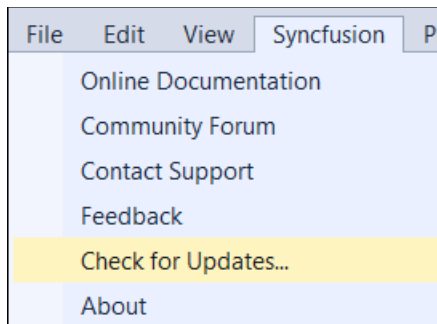
Note: The Syncfusion Check for updates is available from v17.1.0.32.

You can check updates availability in Visual Studio, and then install the update version if required.

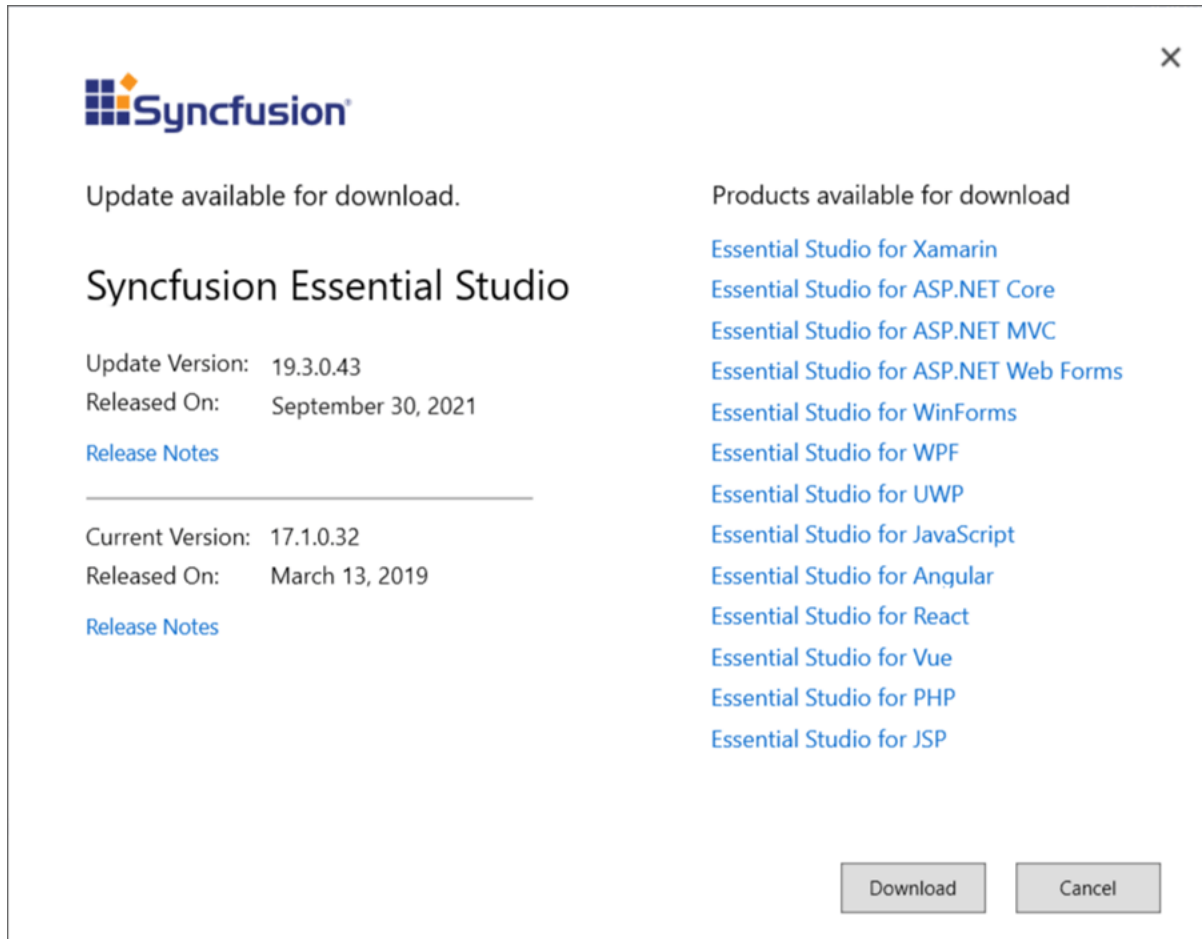
1. Choose **Extensions->Syncfusion -> Check for Updates...** in the Visual Studio menu.



Note: In Visual Studio 2017 or lower, you can see the Syncfusion menu directly in the Visual Studio menu..



2. When there is an update, **Update** dialog box opens.



3. You can download the Syncfusion Essential Studio from the Syncfusion website by selecting **Download**.

Syncfusion Notifications

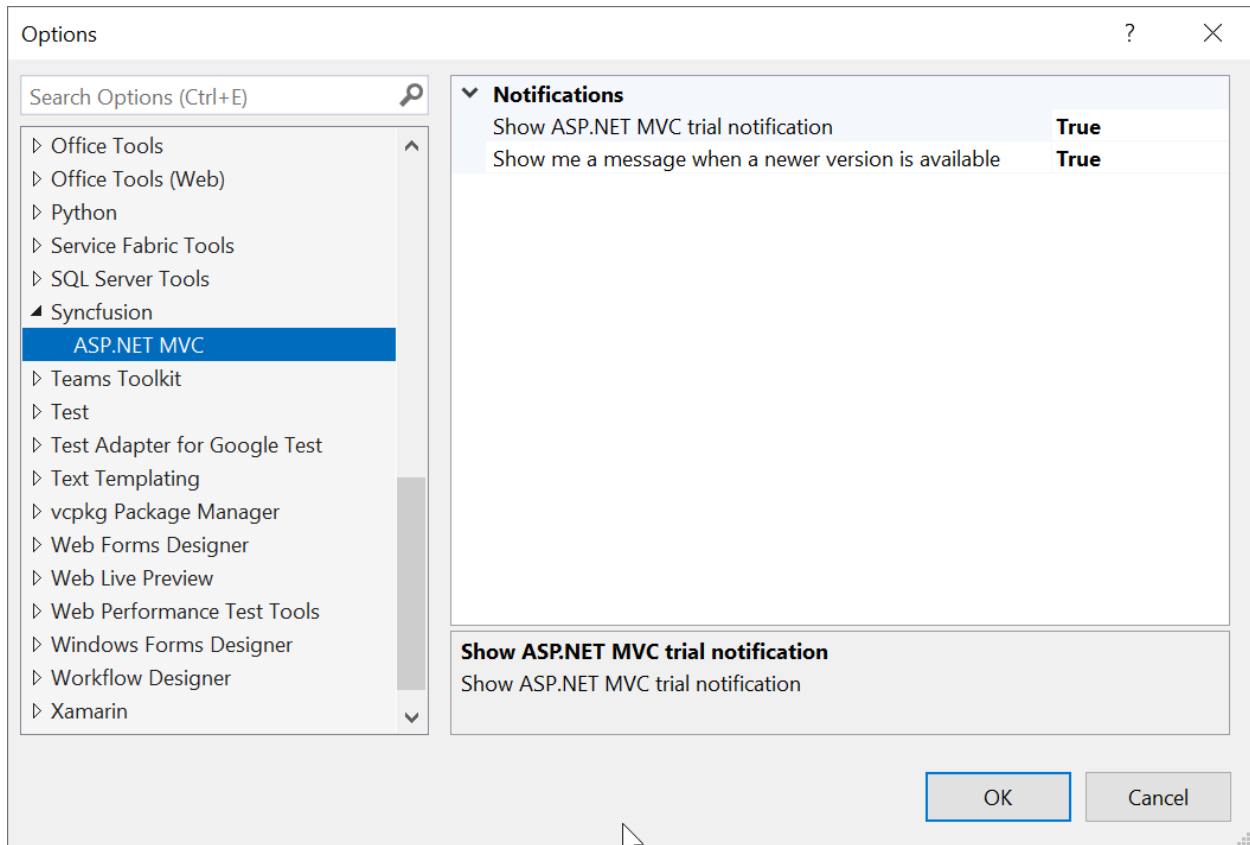
Syncfusion enhances the user experience in ASP.NET MVC applications through notification messages. These notifications cover various aspects, including alerts for trial applications when utilizing Syncfusion trial assemblies, updates regarding the availability of the latest Syncfusion NuGet package, and notifications regarding newer releases of Essential Studio. By keeping users informed, Syncfusion ensures that developers stay updated with Syncfusion latest features and enhancements.

Note: The Syncfusion Notification feature is available from Essential Studio v22.1.34.

Notification Configuration

The Syncfusion Options page allows you to configure notification settings. Customise trial and newer version notifications with a simple true or false toggle.

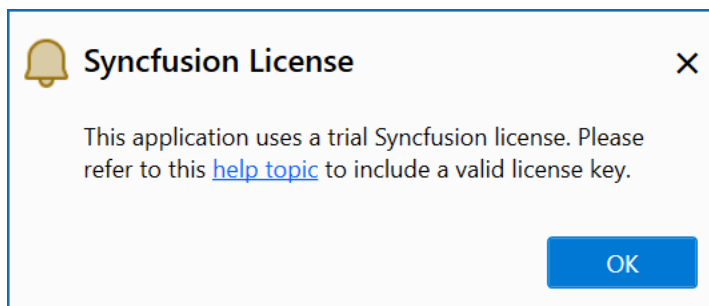
It can be accessed by clicking **Tools -> Options -> Syncfusion -> ASP.NET MVC**



Notification Types

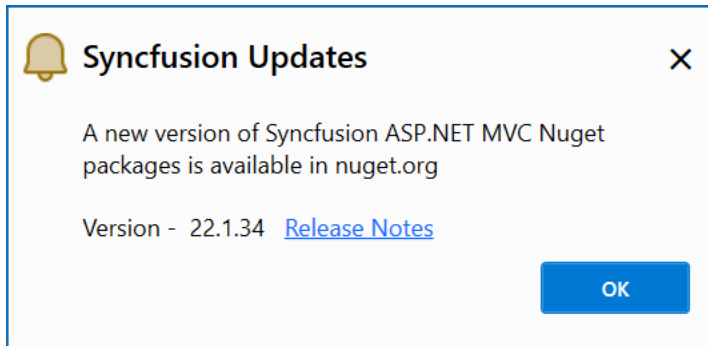
1. Syncfusion Trial Application Notification

When you utilize Syncfusion trial assemblies in your ASP.NET MVC application, you will receive a notification stating, **This application uses a trial Syncfusion license.** This notification encourages you to obtain a valid license key, enabling you to fully explore and experience the extensive features and capabilities offered by Syncfusion.



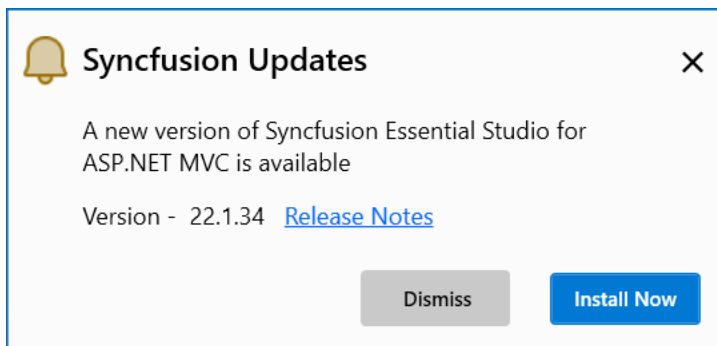
2. Newer Syncfusion NuGet Package Notification

If you have installed lower versions of Syncfusion NuGet packages in your application, you will be notified about the availability of higher versions of Syncfusion NuGet packages on nuget.org. This empowers you to easily identify opportunities to upgrade and gain access to new features, performance enhancements, and bug fixes.



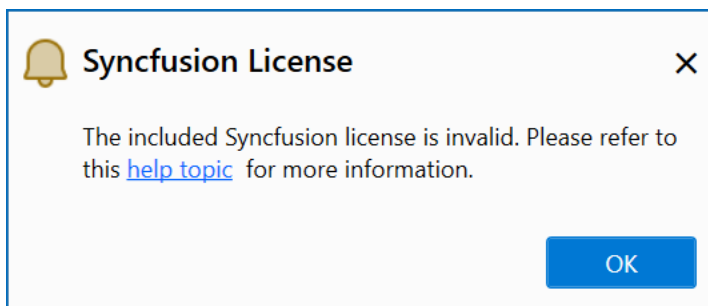
3. Newer Essential Studio Build Notification

If you use older versions of Syncfusion assemblies or NuGet packages from **Essential Studio ASP.NET MVC - EJ2**, Syncfusion will notify you about new releases for the latest Essential Studio build. Updating to the newest version ensures access to recent features, enhancements, and important updates, maximizing the capabilities of Syncfusion in your ASP.NET MVC development projects.



4. Invalid License Key Notification

If you have mistakenly used an incorrect license key or used a license from another version or platform in your ASP.NET MVC application, Syncfusion will display a notification message stating, **The provided Syncfusion license key is invalid**. This message serves as a reminder to obtain a valid license key and ensure proper licensing for Syncfusion components.



Accordion

Getting Started with ASP.NET MVC Accordion Control

This section briefly explains about how to include [ASP.NET MVC Accordion](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it. Alternatively, you can utilize the following package manager command to achieve the same.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

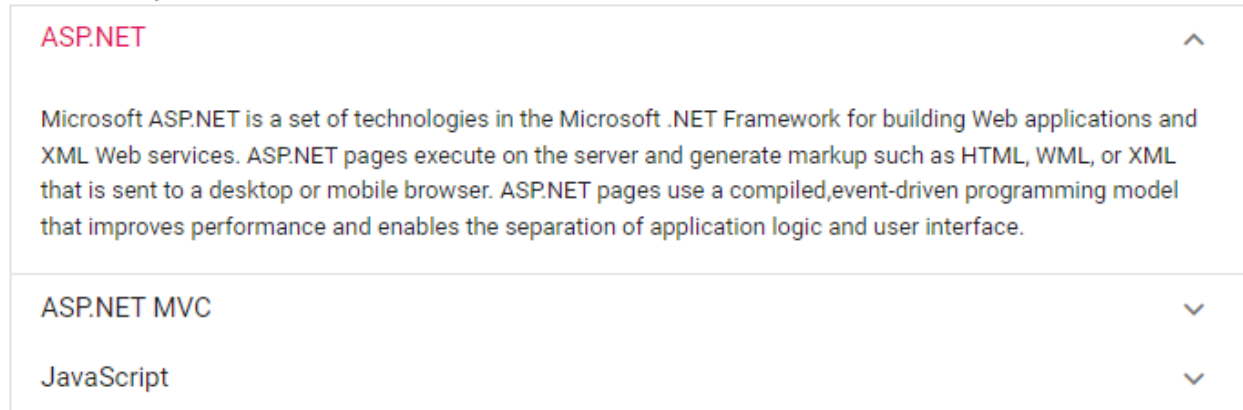
Add ASP.NET MVC Accordion control

Now, add the Syncfusion ASP.NET MVC Accordion control in `~/Home/Index.cshtml` page.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Accordion("defaultAccordion")
    .Items(new List<AccordionItem> {
        new AccordionItem { Header = "ASP.NET", Expanded = true, Content =
"Microsoft ASP.NET is a set of technologies in the Microsoft .NET Framework
for building Web applications and XML Web services. ASP.NET pages execute on
the server and generate markup such as HTML, WML, or XML that is sent to a
desktop or mobile browser. ASP.NET pages use a compiled, event-driven
programming model that improves performance and enables the separation of
application logic and user interface." },
        new AccordionItem { Header = "ASP.NET MVC", Content = "The Model-
View-Controller (MVC) architectural pattern separates an application into
three main components: the model, the view, and the controller. The ASP.NET
MVC framework provides an alternative to the ASP.NET Web Forms pattern for
creating Web applications. The ASP.NET MVC framework is a lightweight,
highly testable presentation framework that (as with Web Forms-based
applications) is integrated with existing ASP.NET features, such as master
pages and membership-based authentication." },
        new AccordionItem { Header = "JavaScript", Content = "JavaScript
(JS) is an interpreted computer programming language. It was originally
implemented as part of web browsers so that client-side scripts could
interact with the user, control the browser, communicate asynchronously, and
alter the document content that was displayed. More recently, however, it has
become common in both game development and the creation of desktop
applications." }
    }).Render()
)
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to launch the app. Then, the Syncfusion ASP.NET MVC Accordion control will be rendered in the default web browser.



Render the Accordion using content template

You can bind any data in Accordion items, by simply using the content template property in ASP.NET Accordion. Accordion is already provided with the content template support and hence this support can be utilized to load the other HTML elements or as per your requirement.

In the below demo, the Accordion items are given with [Chart](#), [Grid](#), [Calender](#) as their content using the content template.

CSHTML

```
@using Syncfusion.EJ2.Charts;
@using Syncfusion.EJ2.Navigations;
@model List<Accordion.Controllers.LineChartData>
<div>
    Content Template
</div>
@(Html.EJS().Accordion("defaultAccordion")
    .ContentTemplate(
        @<div class="e-accordion-container">
            <div>
                <div>
                    <div> ASP.NET MVC Chart </div>
                </div>
                <div>
                    <div>
                        @(Html.EJS().Chart("container")
                            .Series(series => {

series.Type(ChartSeriesType.Line).Width(2).XName("xValue").Marker(mr=>mr.Visible(true).Width(10).Height(10)).YName("yValue").DataSource(Model).Name("Germany").Add();

series.Type(ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(mr =>
mr.Visible(true).Width(10).Height(10)).DataSource(Model).Name("England").Add();

                                })
                                .Title("Inflation - Consumer Price")
                                .Render()
                            )
                        </div>
                    </div>
                </div>
            </div>
        )
    )
    </div>
```

```

        </div>
        <div>
            <div>
                <div>ASP.NET MVC Grid </div>
            </div>
            <div>
                <div>
                    @(Html.EJS().Grid("ej2grid")
                        .Height("400px")
                        .DataSource(dataManger => {

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/Products")
.CrossDomain(true).Adaptor("ODataV4Adaptor");
                    })
                        .Columns(col => {
                            col.Field("ProductID").HeaderText("Product
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
                            col.Field("ProductName").HeaderText("Product
Name").Width("150").Add();
                            col.Field("UnitPrice").HeaderText("Supplier
ID").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("UnitsInStock").HeaderText("QuantityPerUnit").Width("120").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Discontinued").HeaderText("Discontinued").Width("140").TextAlign(
Syncfusion.EJ2.Grids.TextAlign.Center).Type("boolean").DisplayAsCheckBox(tru
e).Add();
                        })
                        .AllowPaging()
                        .Render()
                    )
                </div>
            </div>
        </div>
        <div>
            <div>
                <div>ASP.NET MVC Calendar </div>
            </div>
            <div>
                <div>
                    @(Html.EJS().Calendar("calendar").Render()
                </div>
            </div>
        </div>
    </div>
)
.Render()
)

```

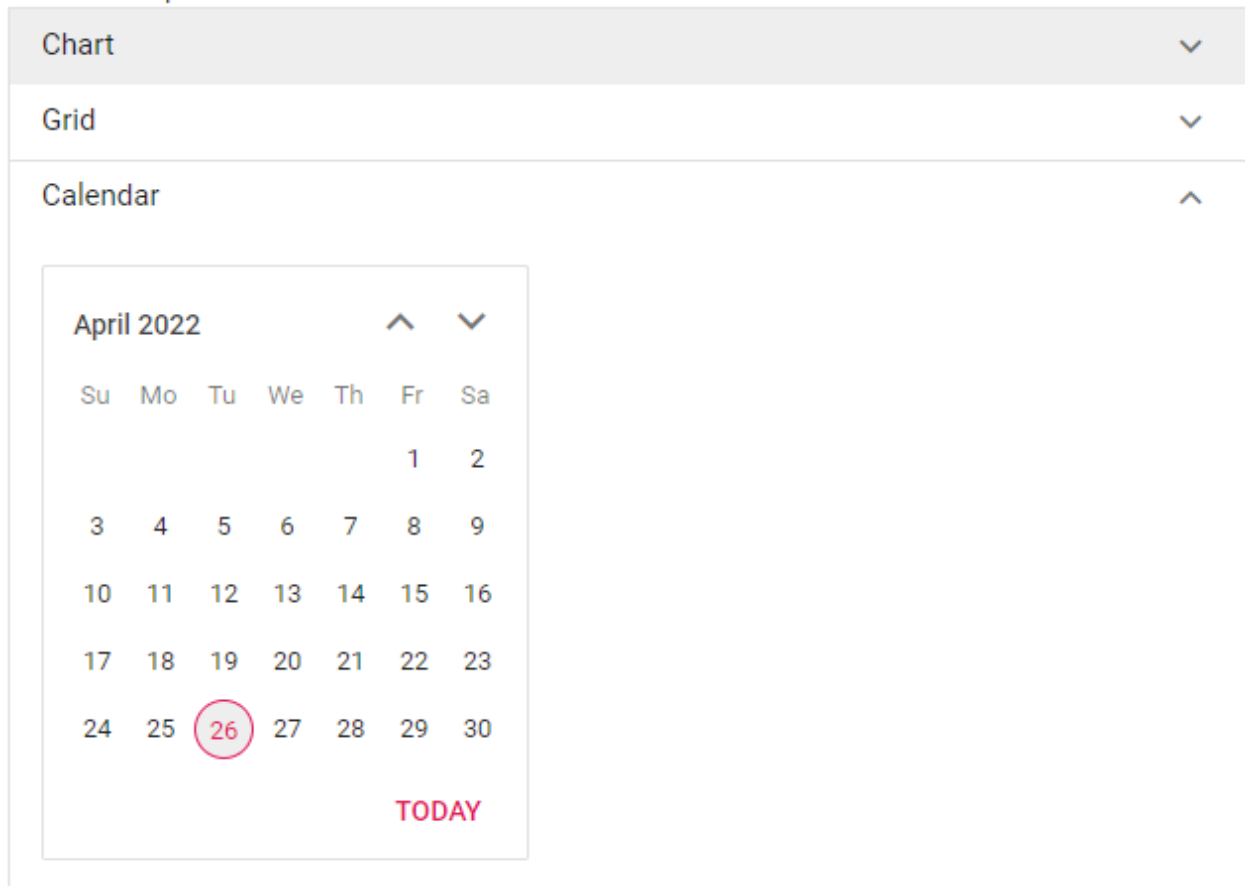
HOMECONTROLLER.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {

```

```
new LineChartData { xValue = new DateTime(2005, 01, 01), yValue = 21,
yValue1 = 28 },
new LineChartData { xValue = new DateTime(2006, 01, 01), yValue = 24,
yValue1 = 44 },
new LineChartData { xValue = new DateTime(2007, 01, 01), yValue = 36,
yValue1 = 48 },
new LineChartData { xValue = new DateTime(2008, 01, 01), yValue = 38,
yValue1 = 50 },
new LineChartData { xValue = new DateTime(2009, 01, 01), yValue = 54,
yValue1 = 66 },
new LineChartData { xValue = new DateTime(2010, 01, 01), yValue = 57,
yValue1 = 78 },
new LineChartData { xValue = new DateTime(2011, 01, 01), yValue = 70,
yValue1 = 84 },
};
return View(chartData);
}
....
....
public class LineChartData
{
public DateTime xValue;
public double yValue;
public double yValue1;
}
```



You can also render accordion without using `ContentTemplate` which can be referred [here](#).

Note: [View Sample in GitHub](#).

Expand Mode in ASP.NET MVC Accordion Control

The Accordion supports the two listed types of expand modes while expanding or collapsing the item.

- Single
- Multiple

Single

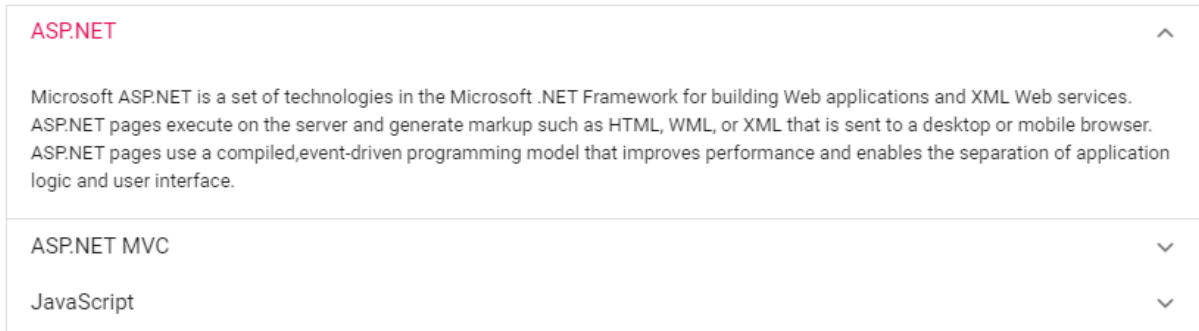
The property enables to expand only one Accordion item at a time. If you expand any new item, the previously expanded one is collapsed and new item changed to expanded state.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().Accordion("defaultAccordion")
    .ExpandMode(ExpandMode.Single)
    .Items(new List<AccordionAccordionItem> {
        new AccordionAccordionItem { Header = "ASP.NET", Expanded = true,
        Content = "Microsoft ASP.NET is a set of technologies in the Microsoft .NET
        Framework for building Web applications and XML Web services. ASP.NET pages
        execute on the server and generate markup such as HTML, WML, or XML that is
        sent to a desktop or mobile browser. ASP.NET pages use a compiled, event-
        driven programming model that improves performance and enables the
        separation of application logic and user interface." },
        new AccordionAccordionItem { Header = "ASP.NET MVC", Content = "The
        Model-View-Controller (MVC) architectural pattern separates an application
        into three main components: the model, the view, and the controller. The
        ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms
        pattern for creating Web applications. The ASP.NET MVC framework is a
        lightweight, highly testable presentation framework that (as with Web Forms-
        based applications) is integrated with existing ASP.NET features, such as
        master pages and membership-based authentication." },
        new AccordionAccordionItem { Header = "JavaScript", Content =
        "JavaScript (JS) is an interpreted computer programming language. It was
        originally implemented as part of web browsers so that client-side scripts
        could interact with the user, control the browser, communicate
        asynchronously, and alter the document content that was displayed. More
        recently, however, it has become common in both game development and the
        creation of desktop applications." }
    })
    .Render()
)
```

SINGLE.CS

```
public ActionResult Index()
{
    return View();
}
```



Multiple

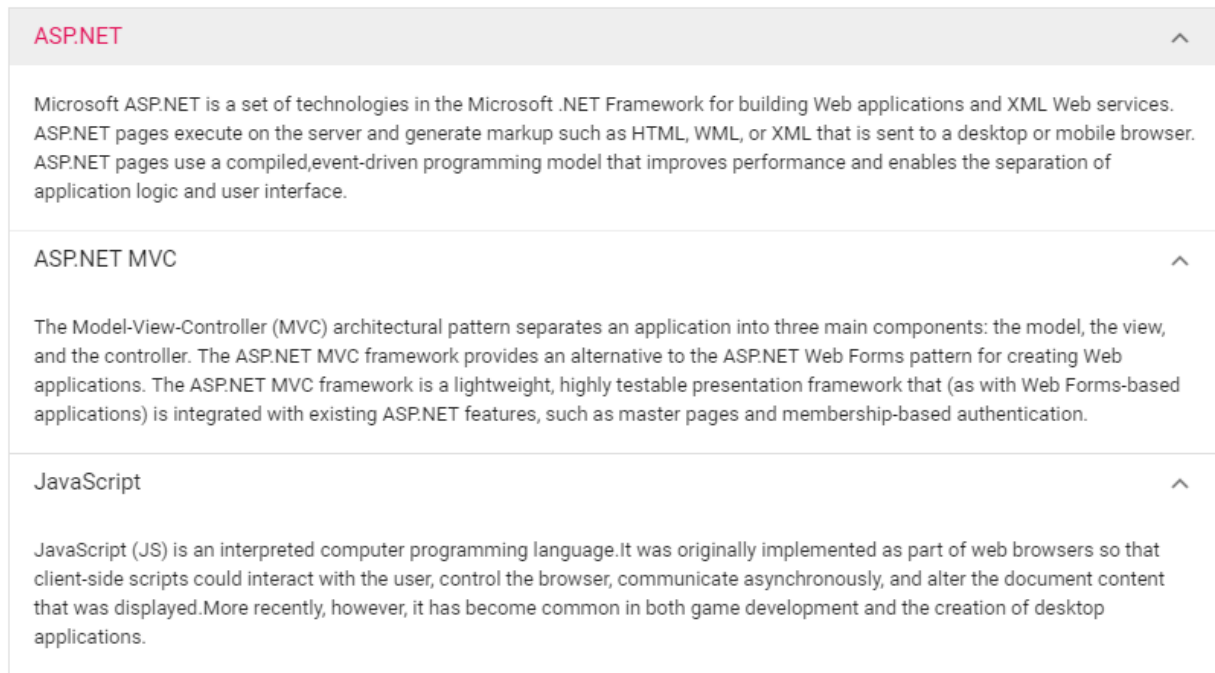
Default expand mode of the Accordion is **Multiple**. It enables you to expand more than one Accordion item at a time. Expand or collapse action can also be toggled by clicking on it again. For example, expanded item is collapsed when you click on it again.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Accordion("defaultAccordion")
    .ExpandMode(ExpandMode.Multiple)
    .Items(new List<AccordionAccordionItem> {
        new AccordionAccordionItem { Header = "ASP.NET", Expanded = true,
        Content = "Microsoft ASP.NET is a set of technologies in the Microsoft .NET Framework for building Web applications and XML Web services. ASP.NET pages execute on the server and generate markup such as HTML, WML, or XML that is sent to a desktop or mobile browser. ASP.NET pages use a compiled, event-driven programming model that improves performance and enables the separation of application logic and user interface." },
        new AccordionAccordionItem { Header = "ASP.NET MVC", Expanded = true, Content = "The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. The ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms pattern for creating Web applications. The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication." },
        new AccordionAccordionItem { Header = "JavaScript", Expanded = true, Content = "JavaScript (JS) is an interpreted computer programming language. It was originally implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. More recently, however, it has become common in both game development and the creation of desktop applications." }
    })
    .Render()
)
```

MULTIPLE.CS

```
public ActionResult Index()
{
    return View();
}
```



Note: [View Sample in GitHub.](#)

See also

- [How to keep single pane open always](#)

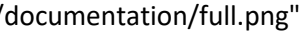
Accessibility in ASP.NET MVC Accordion Control

The Accordion control has been designed keeping in mind the [WAI-ARIA](#) specifications, by applying the prompt WAI-ARIA roles, states, and properties along with the keyboard support. Thus, making it usable for people who use assistive WAI-ARIA Accessibility supports that is achieved through the attributes like `aria-labelledby`. It helps to provides information about the elements in a document for assistive technology. The control implements the keyboard navigation support by following the [WAI-ARIA practices](#) and tested in major screen readers.

The accessibility compliance for the Accordion control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

ARIA attributes

| Roles and Attributes | Functionalities

| role | **Button:** Attribute is set to the Accordion header elements to indicate that the element can be used to toggle the visibility of the associated content section, describing the actual role of the element.
 Region: Attribute is set to the Accordion panel elements to create a landmark region that contains the currently expanded accordion panel, describing the actual role of the element.
|

| aria-labelledby | Attribute is set to content (panel) and it points to the corresponding Accordion header.

| aria-controls | Attribute is set to the header and it points to the corresponding Accordion content.

| aria-expanded | Attribute is set to the Accordion header elements to indicates the expand state of the Accordion Item. Default value of this attribute is `false`. If an item is expanded, the attribute value changes to 'true'. |

| aria-hidden | Attribute is set to the Accordion panel elements to indicates the content visible state of the Accordion Item. Default value of this attribute is `true`. If an item content is visible, the attribute value changes to `false`. |

| aria-disabled | It indicates the disabled state of the Accordion and its items. |

Keyboard interaction

Keyboard navigation is enabled by default. Possible keys are:

Key	Description
----- ----- -----	
Space or Enter	When focus is on the Accordion header, click on the focused element makes the element to expand and collapse.
Down Arrow	Focus the next Accordion header.
Up Arrow	Focus the previous Accordion header.
Home	Focus the first Accordion header.
End	Focus the last Accordion header.
Tab	To Move focus through the interactive elements.
Shift + Tab	To Move focus through the interactive elements.

Ensuring accessibility

The Accordion control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Accordion control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Accordion control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

CSS Structure in Accordion Control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing Accordion

Use the following CSS to customize the Accordion.

```
`CSS
.e-accordion {
border: 5px solid rgb(173, 255, 47);
```

```
}
`
```

Customizing the list items

Use the following CSS to customize the items of Accordion.

```
`CSS
.e-accordion .e-acrdn-item {
text-align: center;
color: pink;
background-color: #2fa1ff;
}
`
```

Customizing Accordion's header

Use the following CSS to customize the header of Accordion control.

```
`CSS
.e-accordion .e-acrdn-item.e-select > .e-acrdn-header {
background: #2fa1ff !important;
justify-content: center;
}
`
```

Customizing Accordion's expand and collapse icons

Use the following CSS to customize the expand and collapse icons of Accordion control.

```
`CSS
.e-accordion .e-acrdn-item .e-acrdn-header .e-toggle-icon .e-icons {
color: pink;
}
`
```

Customizing the hover state of Accordion control

Use the following CSS to customize the accordion item when hovering.

```
`CSS
.e-accordion .e-acrdn-item .e-acrdn-header:hover {
border: 2px solid gray;
}
`
```

Customizing selected item of Accordion control

Use the following CSS to customize the selected accordion item.

`CSS

```
.e-accordion .e-acrdn-item.e-select.e-active>.e-acrdn-header,
.e-accordion .e-acrdn-item.e-select.e-item-focus>.e-acrdn-header {
background-color: rgb(0, 15, 100) !important;
}
`
```

Use the following CSS to customize the selected accordion item text.

`CSS

```
.e-accordion .e-acrdn-item.e-select.e-active>.e-acrdn-header .e-acrdn-header-content,
.e-accordion .e-acrdn-item.e-select.e-item-focus>.e-acrdn-header .e-acrdn-header-content {
color: #2fa1ff !important;
}
`
```

Note: [View Sample in GitHub.](#)

How To

Set the nested Accordion

Accordion supports to render **nested** level of Accordion by using content property. You can give nested Accordion content inside the parent Accordion content property by using **id** of nested element.

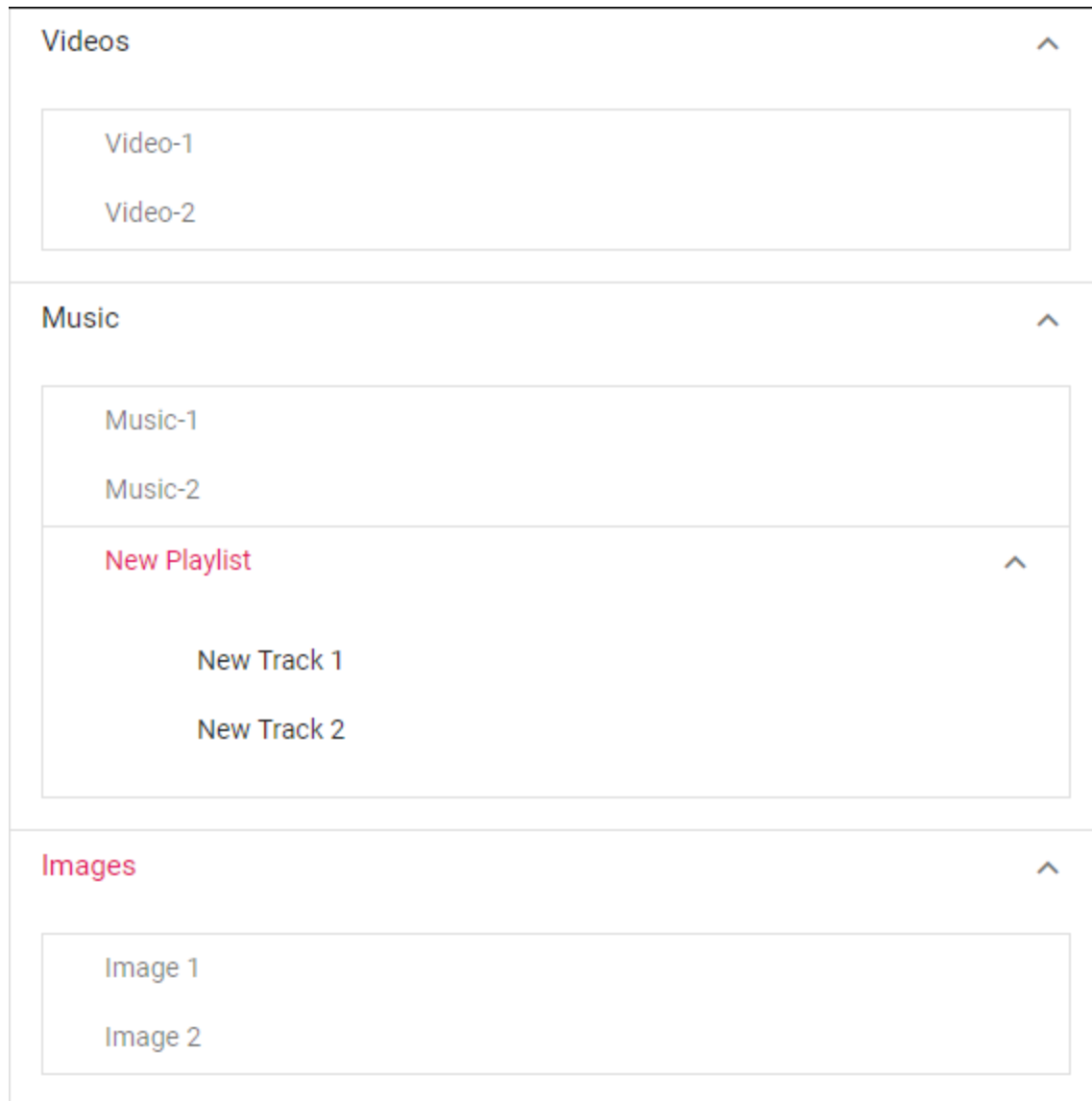
CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div style="display: none">
    @Html.EJS().Accordion("childNested1").Items(new
List<AccordionAccordionItem>
    {
        new AccordionAccordionItem { Header = "Video-1" },
        new AccordionAccordionItem { Header = "Video-2" }
    }).Render()
</div>
<div style="display: none">
    @Html.EJS().Accordion("childNested2").Items(new
List<AccordionAccordionItem>
    {
        new AccordionAccordionItem { Header = "Music-1" },
        new AccordionAccordionItem { Header = "Music-2" },
        new AccordionAccordionItem { Header = "New Playlist", Expanded =
true, Content= "#childNested4" }
    }).Render()
</div>
<div style="display: none">
    @Html.EJS().Accordion("childNested3").Items(new
List<AccordionAccordionItem>
```

```
{
    new AccordionAccordionItem { Header = "Image-1" },
    new AccordionAccordionItem { Header = "Image-2" }
}).Render()
</div>
<div style="display: none">
    @Html.EJS().Accordion("childNested4").Items(new
List<AccordionAccordionItem>
    {
        new AccordionAccordionItem { Header = "New Track-1" },
        new AccordionAccordionItem { Header = "New Track-2" }
    }).Render()
</div>
<div>
    @Html.EJS().Accordion("ParentNested").Width("50%").ExpandMode(ExpandMode.Multiple).Items(new List<AccordionAccordionItem>
    {
        new AccordionAccordionItem { Header = "Videos", Expanded = true,
Content = "#childNested1" },
        new AccordionAccordionItem { Header = "Music", Expanded = true,
Content = "#childNested2" },
        new AccordionAccordionItem { Header = "Images", Expanded = true,
Content = "#childNested3" }
    }).Render()
</div>
```

NESTED.CS

```
public ActionResult Index()
{
    return View();
}
```



Note: [View Sample in GitHub.](#)

Load content through Ajax

Accordion supports to load external contents through **AJAX** library. Refer the below steps.

- Import the **Ajax** module from **ej2-base** and initialize with URL path.
- Get data from the Ajax Success event to initialize Accordion with retrieved external path data.

CSHTML

```
<div id="element"></div>
<div id="acrdnContnet1" style="display:none">
  <ul style="margin : 0px;padding:0px 16px; list-style-type: none">
    <li>Testing</li>
    <li>Development</li>
  </ul>
</div>
```

```

<div id="acrdnContnet2" style="display:none">
    <ul style="margin : 0px;padding:0px 16px; list-style-type: none">
        <li>Mobile</li>
        <li>Web</li>
    </ul>
</div>
<script>
    var ajax = new ej.base.Ajax('../Scripts/accordion/ajax.html', 'GET',
true);
    ajax.send().then();
    ajax.onSuccess = function (data) {
        var ctn2 = data;
        //Initialize Accordion component
        var accordion = new ej.navigations.Accordion({
            items: [
                { header: 'Department', content: '#acrdnContnet1' },
                { header: 'Platform', content: '#acrdnContnet2' },
                { header: 'Employee Details', content: ctn2 },
            ]
        });
        //Render initialized Accordion component
        accordion.appendTo('#element');
    }
</script>
<style>
    .accordion-control-section {
        margin: 0 10% 0 10%;
    }
    @@media screen and (max-width: 768px) {
        .accordion-control-section {
            margin: 0;
        }
    }
</style>

```

AJAX.CS

```

public ActionResult Index()
{
    return View();
}

```

To keep single pane open always

By default, all Accordion panels are collapsible. You can customize the Accordion to keep one panel as expanded state always. This is applicable for **Single** expand mode.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().Accordion("defaultAccordion")
    .Clicked("clicked")
    .Expanding("beforeExpand")
    .Items(new List<AccordionAccordionItem> {

```

```

        new AccordionAccordionItem { Header = "ASP.NET", Expanded = true,
Content = "Microsoft ASP.NET is a set of technologies in the Microsoft .NET
Framework for building Web applications and XML Web services." },
        new AccordionAccordionItem { Header = "ASP.NET MVC", Content = "The
Model-View-Controller (MVC) architectural pattern separates an application
into three main components: the model, the view, and the controller." },
        new AccordionAccordionItem { Header = "JavaScript", Content =
"JavaScript (JS) is an interpreted computer programming language.It was
originally implemented as part of web browsers so that client-side scripts
could interact with the user, control the browser, communicate
asynchronously, and alter the document content that was displayed." }
    })
    .Render()
)
<script type="text/javascript">
    var clickEle;
    function clicked(e) {
        clickEle = e.originalEvent.target.closest('.e-acrdn-header');
    }
    function beforeExpand(e) {
        var expandCount = this.element.querySelectorAll('.e-
selected').length;
        var ele = this.element.querySelectorAll('.e-selected')[0];
        var tglIcon = ele;
        if (ele) {
            ele = ele.firstChild;
            tglIcon = ele.querySelector('.e-tgl-collapse-icon');
        }
        if (expandCount === 1 && (ele === clickEle || tglIcon === clickEle))
    {
        e.cancel = true;
    }
    }
}
</script>

```

OPENALWAYS.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Create wizard using Accordion

Accordion items can be disabled dynamically by passing the index and boolean value with the enableItem method.

The below demo is designed for simple payment module that enable or disable Accordion based on sequential validation of each Accordion content.

CSHTML

```

@using Syncfusion.EJ2.Navigations
@using Syncfusion.EJ2.Inputs
<div class="accordion-control-section">

```



```

<div class="accordion-control-section">
  <div class='template_title'> Online Shopping Payment Module</div>
  @(Html.EJS().Accordion("element")
    .Expanding("expand")
    .Created("created")
    .ContentTemplate(
      @<div class="e-accordion-container">
        <div>
          <div>
            Sign In
          </div>
          <div>
            <div id="Sign_In_Form" style="padding: 10px">
              <form id="formId">
                <div class="form-group">
                  <div class="e-float-input">
                    <input type="text" id="email"
name="Email" required="" />
                    <span class="e-float-
line"></span>
                    <label class="e-float-text"
for="email">Email</label>
                  </div>
                  <div class="e-float-input">
                    <input class="e-input"
id="password" type="password" name="Password" required="" />
                    <span class="e-float-
line"></span>
                    <label class="e-float-text"
for="password">Password</label>
                  </div>
                </div>
              </form>
              <div style="text-align: center">
                <button class='e-btn'
id="Continue_Btn">Continue</button>
                <div id="err1">* Please fill all
fields</div>
              </div>
            </div>
          </div>
        </div>
        <div>
          <div>
            Delivery Address
          </div>
          <div>
            <div id="Address_Fill" style="padding: 10px">
              <form id="formId_Address">
                <div class="form-group">
                  <div class="e-float-input">
                    <input type="text" id="name"
name="Name" required="" />
                    <span class="e-float-
line"></span>
                    <label class="e-float-text"
for="name">Name</label>

```

```

        </div>
    </div>
    <div class="form-group">
        <div class="e-float-input">
            <input type="text" id="address"
name="Address" required="" />
            <span class="e-float-
line"></span>
            <label class="e-float-text"
for="address">Address</label>
        </div>
    </div>
    <div class="form-group">
        @Html.EJS().NumericTextBox("mobile").Placeholder("Mobile").FloatLabelType(Fl
oatLabelType.Auto).Format("0").ShowSpinButton(false).Render()
        </div>
    </form>
    <div style="text-align: center">
        <button class='e-btn'
id="Continue_BtnAdr">Continue</button>
        <div id="err2">* Please fill all
fields</div>
    </div>
</div>
</div>
</div>
<div>
    <div>
        Card Details
    </div>
    <div>
        <div id="Card_Fill" style="padding: 10px">
            <div class="col-xs-6 col-sm-6 col-lg-6 col-
md-6">
                <div class="form-group">
                    @Html.EJS().NumericTextBox("cardNo").Placeholder("Card
No").FloatLabelType(FloatLabelType.Auto).Format("0").ShowSpinButton(false).R
ender()
                    </div>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-
md-6">
                    <div class="form-group">
                        <div class="e-float-input">
                            <input type="text"
id="cardHolder" name="cardHolder" required="" />
                            <span class="e-float-
line"></span>
                            <label class="e-float-text"
for="cardHolder">CardHolder Name</label>
                        </div>
                    </div>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-
md-6">

```

```

@Html.EJS().DatePicker("expiry").Readonly(false).Format("MM/yyyy").Placeholder("Expiry Date").Width("100%").Render()
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
    <div class="form-group">
@Html.EJS().NumericTextBox("CVV").Placeholder("CVV").FloatLabelType(FloatLabelType.Auto).Format("0").ShowSpinButton(false).Render()
    </div>
</div>
<div style="text-align: center">
    <button class='e-btn'
id="Back_Btn">Back</button>
    <button class='e-btn'
id="Save_Btn">Save</button>
    <div id="err3">* Please fill all
fields</div>
    </div>
</div>
</div>
</div>
    ).Render()
)
</div>
<div>
@Html.EJS().Dialog("alertDialog").Header("Alert").Target("#element").ShowCloseIcon(true).Width("250").IsModal(true).Created("dlgCreated").Content("").Visible(false).Render()
    </div>
</div>
<style>
    .accordion-control-section {
        margin: 0 10% 0 10%;
    }
    @@media screen and (max-width: 768px) {
        .accordion-control-section {
            margin: 0;
        }
    }
    #err1, #err2, #err3 {
        display: none;
        color: red;
        margin-top: 10px;
        font-weight: 500;
    }
    #err1.show,
    #err2.show,
    #err3.show {
        display: block;
    }
    .e-dialog {
        max-height: 300px !important;
    }

```

```

        .template_title {
            text-align: center;
            padding: 10px 0;
            margin: 20px 0;
            text-overflow: ellipsis;
            font-weight: bold;
            font-size: 16px;
        }
    </style>
    <script>
        var success = 'Your payment successfully processed';
        var email_alert = 'Enter valid email address';
        var mobile_alert = 'Mobile number length should be 10';
        var card_alert = 'Card number length should be 16';
        var cvv_alert = 'CVV number length should be 3';
        var alertDlg;
        var acrdnObj;
        function dlgCreated() {
            alertDlg = document.getElementById("alertDialog").ej2_instances[0];
            alertDlg.buttons = [{
                buttonModel: { content: "Ok", isPrimary: true },
                click: function () {
                    alertDlg.hide();
                    if (acrdnObj.expandedIndices[0] === 2 && alertDlg.content
=== success) {
                        acrdnObj.enableItem(0, true);
                        acrdnObj.enableItem(1, false);
                        acrdnObj.enableItem(2, false);
                        acrdnObj.expandItem(true, 0);
                    }
                }
            }];
            alertDlg.dataBind();
            alertDlg.hide();
        }
        function checkMail(mail) {
            if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/ .test(mail)) {
                return (true);
            } else {
                alertDlg.content = email_alert;
                alertDlg.show();
                return (false);
            }
        }
        function checkMobile(mobile) {
            if (mobile.match(/^\d{10}$/)) {
                return (true);
            } else {
                alertDlg.content = mobile_alert;
                alertDlg.show();
                return (false);
            }
        }
        function checkCardNo(cardNo) {
            if (cardNo.match(/^\d{16}$/)) {
                return (true);
            } else {

```

```

        alertDlg.content = card_alert;
        alertDlg.show();
        return (false);
    }
}
function checkCVV(cvv) {
    if (cvv.match(/^\\d{3}$\\/)) {
        return (true);
    } else {
        alertDlg.content = cvv_alert;
        alertDlg.show();
        return (false);
    }
}
function created(e) {
    acrdnObj = document.getElementById("element").ej2_instances[0];
    acrdnObj.enableItem(1, false);
    acrdnObj.enableItem(2, false);
}
function expand(e) {
    if (e.name === 'expanding' && (e.index === 0)) {
        document.getElementById('Continue_Btn').onclick = (e) => {
            var email = document.getElementById('email');
            var password = document.getElementById('password');
            if (email.value !== '' && password.value !== '') {
                if (checkMail(email.value)) {
                    email.value = password.value = '';
                    acrdnObj.enableItem(1, true);
                    acrdnObj.enableItem(0, false);
                    acrdnObj.expandItem(true, 1);
                }
            }
        };
    } else if (e.name === 'expanding' && (e.index === 1)) {
        document.getElementById('Continue_BtnAdr').onclick = (e) => {
            var name = document.getElementById('name');
            var address = document.getElementById('address');
            var mobile = document.getElementById('mobile');
            if ((name.value !== '') && (address.value !== '') &&
(mobile.value !== '')) {
                if (checkMobile(mobile.value)) {
                    acrdnObj.enableItem(2, true);
                    acrdnObj.enableItem(1, false);
                    acrdnObj.expandItem(true, 2);
                }
            }
        };
    } else if (e.name === 'expanding' && (e.index === 2)) {
        document.getElementById('Back_Btn').onclick = (e) => {

```

```

        acrdnObj.enableItem(1, true);
        acrdnObj.enableItem(2, false);
        acrdnObj.expandItem(true, 1);
    };
    document.getElementById('Save_Btn').onclick = (e) => {
        var cardHolder = document.getElementById('cardHolder');
        var expiry = document.getElementById('expiry');
        var cardNo = document.getElementById('cardNo');
        var cvv = document.getElementById('CVV');
        if ((cardNo.value !== '') && (cardHolder.value !== '') &&
(expiry.value !== '') && (cvv.value !== '')) {
            if (checkCardNo(cardNo.value)) {
                if (checkCVV(cvv.value)) {
                    alertDlg.content = success;
                    alertDlg.show();
                }
            }
        }

        document.getElementById('err3').classList.remove('show');
    } else {
        document.getElementById('err3').classList.add('show');
    }
};
    }
}
</script>

```

WIZARD.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Load accordion with DataSource

You can bind any data object to Accordion items, by mapping it to [header](#) and [content](#) property.

In the below demo, Data is fetched from an OData service using [DataManager](#). The result data is formatted as a JSON object with [header](#) and [content](#) fields, which is set to items property of Accordion.

CSHTML

```

<div id='element'></div>
<script>
var itemsData = [];
var mapping = { header: 'FirstName', content: 'Notes' };
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Employees';
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().range(4, 7)).then(function (e) {
        var result = e.result;
        for (var i = 0; i < result.length; i++) {

```

```

        itemsData.push({ header: result[i][mapping.header], content:
result[i][mapping.content] });
    }
    //Initialize Accordion component
    var accordion = new ej.navigations.Accordion({
        items: itemsData
    });
    //Render initialized Accordion component
    accordion.appendTo('#element');
});
</script>
<style>
    .accordion-control-section {
        margin: 0 10% 0 10%;
    }
    @media screen and (max-width: 768px) {
        .accordion-control-section {
            margin: 0;
        }
    }
</style>

```

DATASOURCE.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Customize Accordion expand or collapse animation behavior

Accordion component supports customizing the expand or collapse animation action behavior. You can manually adjust the expand animation action that occurs after the collapse animation operation is performed on the already expanded pane, When the expand icons are clicked.

By default, the Accordion component pane is expanded or collapsed, while clicking the expand or collapse icon. It is not affected on already expanded pane.

The following sample demonstrates, how to expand the collapsed Accordion item after collapse animation is performed on the expanded Accordion item using `created`, `expanding`, and `expanded` event. In the `Expanding` event, get the previously expanded item index and prevent the expanding behavior using `args.cancel` option. Expand the Accordion item dynamically based on the specified `index` value using the `expandItem` public method and `expanded` event.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().Accordion("defaultAccordion")
    .ExpandMode(ExpandMode.Single)
    .Created("created")
    .Expanded("expanded")
    .Expanding("expanding")
    .Items(new List<AccordionAccordionItem> {

```

```

        new AccordionAccordionItem { Header = "ASP.NET", Expanded = true,
Content = "Microsoft ASP.NET is a set of technologies in the Microsoft .NET
Framework for building Web applications and XML Web services. ASP.NET pages
execute on the server and generate markup such as HTML, WML, or XML that is
sent to a desktop or mobile browser. ASP.NET pages use a compiled,event-
driven programming model that improves performance and enables the
separation of application logic and user interface." },
        new AccordionAccordionItem { Header = "ASP.NET MVC", Content = "The
Model-View-Controller (MVC) architectural pattern separates an application
into three main components: the model, the view, and the controller. The
ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms
pattern for creating Web applications. The ASP.NET MVC framework is a
lightweight, highly testable presentation framework that (as with Web Forms-
based applications) is integrated with existing ASP.NET features, such as
master pages and membership-based authentication." },
        new AccordionAccordionItem { Header = "JavaScript", Content =
"JavaScript (JS) is an interpreted computer programming language.It was
originally implemented as part of web browsers so that client-side scripts
could interact with the user, control the browser, communicate
asynchronously, and alter the document content that was displayed.More
recently, however, it has become common in both game development and the
creation of desktop applications." }
    })
    .Render()
)
<script>
    var initialLoad = true;
    var isCollapsed = false;
    var expandIndex;
    var tabEle = document.getElementById("defaultAccordion");
    function expanding(e) {
        if (e.isExpanded && !initialLoad && !isCollapsed) {
            e.cancel = true;
            expandIndex = tabEle.ej2_instances[0].items.indexOf(e.item);
            isCollapsed = true;
        }
    }
    function expanded(e) {
        if (!e.isExpanded && !initialLoad && isCollapsed) {
            tabEle.ej2_instances[0].expandItem(true, expandIndex);
            isCollapsed = false;
        }
    }
    function created(e) {
        initialLoad = false;
    }
</script>

```

ACTIONS.CS

```

public ActionResult Index()
{
    return View();
}

```


ASP.NET	▼
ASP.NET MVC	▲
<p>The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. The ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms pattern for creating Web applications. The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication.</p>	
JavaScript	▼

Note: [View Sample in GitHub.](#)

Load the content as partial view to Accordion

Since Accordion is a Navigation control, it doesn't have support to load any content directly or using any DataAdapter. But it is provided with the items support. So to load the content as partial view, you would need to make use of the AJAX or EJ2 Datamanager as described in [How-To](#) section help document.

The below demo explains how to create the Accordion items dynamically and then to load the other Syncfusion controls in it from partial views.

CSHTML

```
<div class="control-section">
    <div class="control_wrapper accordion-control-section">

@Html.EJS().Accordion("MainAccordion").Created("Created").Expanding("Expanding").Render()
    </div>
</div>
<script>
    var loaded = false;
    function Created(e) {
        var AccObj =
document.getElementById("MainAccordion").ej2_instances[0];
        var ajax = new ej.base.Ajax('@Url.Action("PartialView1",
"Accordion")', 'GET', true);
        ajax.send().then();
        ajax.onSuccess = function (data) {
            AccObj.addItem({ header: 'Grid1' , content: "<br/><div
id='GridOrder1'></div>", expanded: true },0);
            AccObj.addItem({ header: 'Grid2' , content: "<br/><div
id='GridOrder2'></div>" },1);
            $("#GridOrder1").html(data);
        }
    }
    function Expanding(e) {
        if(e.index != 0 && !loaded){
            var AccObj =
document.getElementById("MainAccordion").ej2_instances[0];
            var ajax = new ej.base.Ajax('@Url.Action("PartialView2",
"Home")', 'GET', true);
```

```

        ajax.send().then();
        ajax.onSuccess = function (data) {
            $("#GridOrder2").html(data);
            loaded = true;
        }
    }
}
</script>
//Code in the PartialView
@Html.EJS().Grid("Grid").Height(250).DataSource(dataManger =>
{
    dataManger.Url("https://ej2services.syncfusion.com/production/web-
services/api/Orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("160").Add();
    col.Field("EmployeeID").HeaderText("Employee
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Freight").HeaderText("Freight").Width("150").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().PageSettings(page=>page.PageCount(3)).Render()
@Html.EJS().ScriptManager()

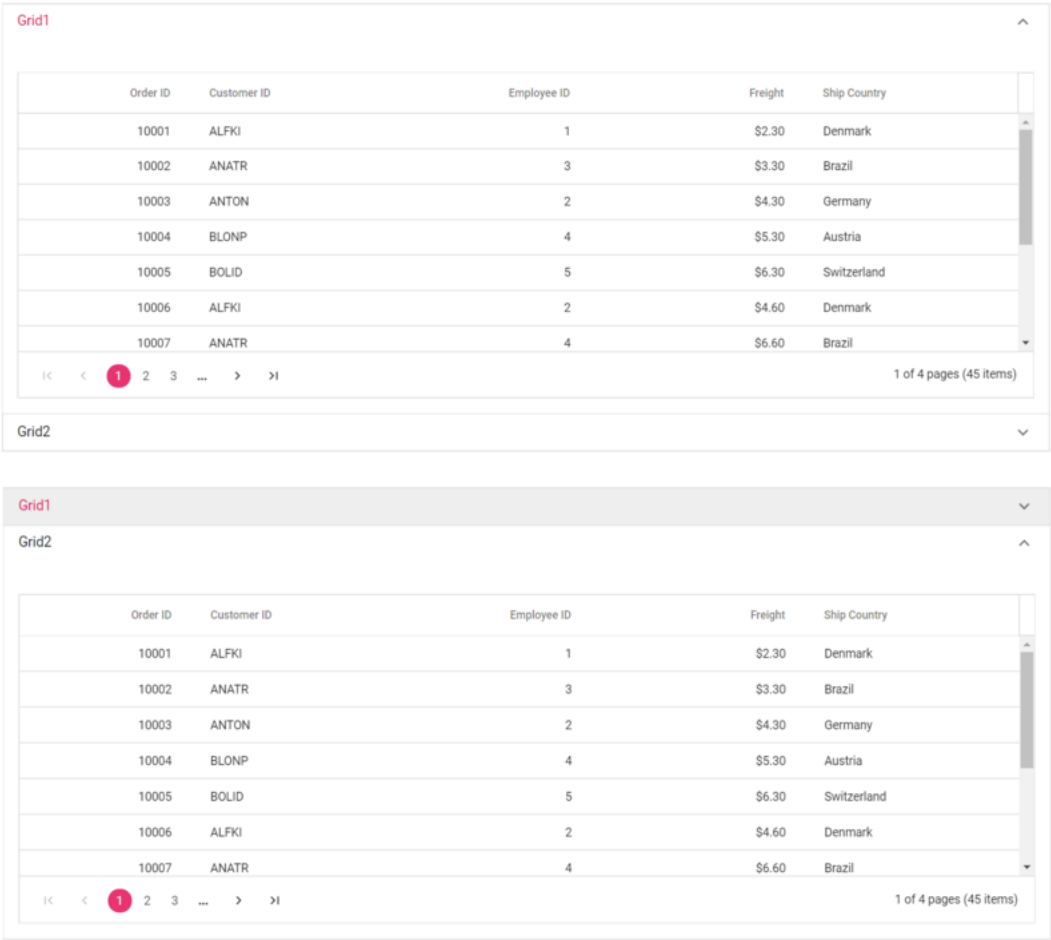
```

PARTIALVIEW.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class AccordionController : Controller
    {
        // GET: //
        public IActionResult DefaultFunctionalities()
        {
            return View();
        }
        public ActionResult PartialView1()
        {
            return PartialView();
        }
        public ActionResult PartialView2()
        {
            return PartialView();
        }
    }
}

```



Migration from Essential JS 1

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Accessibility

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Keyboard Navigation | **Property:** *AllowKeyboardNavigation*

@Html.EJ().Accordion("ejAccoordion").AllowKeyboardNavigation(true).Render()
 | **Not Applicable** |

| Localization | **Not Applicable** | **Property:** *Locale*

@Html.EJS().Accordion("ej2Accordion").Locale("en-US").Render()
 |

| RTL | **Property:** *EnableRTL*

@Html.EJ().Accordion("ejAccoordion").EnableRTL(true).Render()
 | **Property:** *EnableRtl*

> @Html.EJS().Accordion("ej2Accordion").EnableRtl(true).Render()
 |

AjaxSettings

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *AjaxSettings*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.Type("GET");
 }).Render()
 | **Not Applicable** |

| Asynchronous | **Property:** *AjaxSettings.Async*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.Async(true);
 }).Render()
 | **Not Applicable** |

| Browser Cache | **Property:** *AjaxSettings.Cache*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.Cache(false);
 }).Render()
 | **Not Applicable** |

| Request type | **Property:** *AjaxSettings.ContentType*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.ContentType("html");
 }).Render()
 | **Not Applicable** |

| Data | **Property:** *AjaxSettings.Data*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.Data("");
 }).Render()
 | **Not Applicable** |

| Response type | **Property:** *AjaxSettings.DataType*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.DataType("html");
 }).Render()
 | **Not Applicable** |

| HTTP request type | **Property:** *AjaxSettings.Type*

 @Html.EJ().Accordion("ejAccordion").AjaxSettings(ajax => {
 ajax.Type("GET");
 }).Render()
 | **Not Applicable** |

| AjaxBeforeLoad | **Event:** *AjaxBeforeLoad*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e => e.AjaxBeforeLoad("onAjaxBeforeLoad")).Render()
 | **Not Applicable** |

| AjaxError | **Event:** *AjaxError*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e => e.AjaxError("onError")).Render()
 | **Not Applicable** |

| AjaxLoad | **Event:** *AjaxLoad*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e => e.AjaxLoad("onAjaxLoad")).Render()
 | **Not Applicable** |

| AjaxSuccess | **Event:** *AjaxSuccess*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e => e.AjaxSuccess("onAjaxSuccess")).Render()
 | **Not Applicable** |

Animation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Not Applicable** | **Property:** *Animation*

 @Html.EJS().Accordion("ej2Accordion").Animation(anim => {
 anim.Expand("").Collapse("");
 }).Render()
 |

| EnableAnimation | **Property:** *EnableAnimation*

 @Html.EJ().Accordion("ejAccoordion").EnableAnimation(false).Render()
 | **Not Applicable** |

| Expand animation | **Not Applicable** | **Property:** *Animation.Expand*

 @Html.EJS().Accordion("ej2Accordion").Animation(anim => {
 anim.Expand(new

```
List<AccordionAccordionActionSettings>() { <br /> &#160; &#160; new
AccordionAccordionActionSettings() { Effect = "SlideDown" } <br /> &#160; }; <br /> }).Render() <br /> |
| Collapse animation | Not Applicable | Property: Animation.Collapse <br /><br />
@Html.EJS().Accordion("ej2Accordion").Animation(anim => { <br /> &#160; anim.Collapse(new
List<AccordionAccordionActionSettings>() { <br /> &#160; &#160; new
AccordionAccordionActionSettings() { Effect = "SlideUp" } <br /> &#160; }; <br /> }).Render() <br /> |
| Duration <br /> [expand / collapse] | Not Applicable | Property: Animation.Collapse.Duration <br /><br />
@Html.EJS().Accordion("ej2Accordion").Animation(anim => { <br /> &#160; anim.Collapse(new
List<AccordionAccordionActionSettings>() { <br /> &#160; &#160; new
AccordionAccordionActionSettings() { Duration = 400 } <br /> &#160; }; <br /> }).Render() <br /> |
| Easing <br /> [expand / collapse] | Not Applicable | Property: Animation.Collapse.Easing <br /><br />
@Html.EJS().Accordion("ej2Accordion").Animation(anim => { <br /> &#160; anim.Collapse(new
List<AccordionAccordionActionSettings>() { <br /> &#160; &#160; new
AccordionAccordionActionSettings() { Easing = "ease-in" } <br /> &#160; }; <br /> }).Render() <br /> |
| Effect <br /> [expand / collapse] | Not Applicable | Property: Animation.Collapse.Effect <br /><br />
@Html.EJS().Accordion("ej2Accordion").Animation(anim => { <br /> &#160; anim.Collapse(new
List<AccordionAccordionActionSettings>() { <br /> &#160; &#160; new
AccordionAccordionActionSettings() { Effect = "SlideDown" } <br /> &#160; }; <br /> }).Render() <br /> |
```

Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
| Default | Not Applicable | Property: Items <br /><br />
@Html.EJS().Accordion("ej2Accordion").Items(item => { <br /> &#160;
item.Header("").Content("").Add(); <br /> }).Render() <br /> |

| Content | Not Applicable | Property: Items[0].Content <br /><br />
@Html.EJS().Accordion("ej2Accordion").Items(item => { <br /> &#160; item.Content("contents").Add();
<br /> }).Render() <br /> |

| Custom class | Not Applicable | Property: Items[0].CssClass <br /><br />
@Html.EJS().Accordion("ej2Accordion").Items(item => { <br /> &#160;
item.CssClass("customClass").Add(); <br /> }).Render() <br /> |

| Header | Not Applicable | Property: Items[0].Header <br /><br />
@Html.EJS().Accordion("ej2Accordion").Items(item => { <br /> &#160; item.Header("Header").Add(); <br />
/> }).Render() <br /> |

| HeaderSize | Property: HeaderSize <br /><br />
/>@Html.EJ().Accordion("ejAccoordion").HeaderSize("50px").Render() <br /> | Not Applicable |

| Icon class | Not Applicable | Property: Items[0].IconCss <br /><br />
@Html.EJS().Accordion("ej2Accordion").Items(item => { <br /> &#160; item.iconCss("e-icon").Add(); <br />
/> }).Render() <br /> |

| IsExpand | Not Applicable | Property: Items[0].Expanded <br /><br />
@Html.EJS().Accordion("ej2Accordion").Items(item => { <br /> &#160; item.Expand(true).Add(); <br />
}).Render() <br /> |
```

| Collapse Item | **Method:** *collapsePanel(index)*

var accordion =
\$("#accordion").data("ejAccordion");
 accordion.collapsePanel(0);
 | **Method:**
expandItem(index, false)

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.expandItem(0, false);
 |

| Expand Item | **Method:** *expandPanel(index)*

 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.expandPanel(0);
 | **Method:**
expandItem(index, true)

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.expandItem(0, true);
 |

| CollapseAll | **Method:** *collapseAll()*

 var accordion = \$("#accordion").data("ejAccordion");

 accordion.collapseAll();
 | **Not Applicable** |

| ExpandAll | **Method:** *expandAll()*

 var accordion = \$("#accordion").data("ejAccordion");

 accordion.expandAll();
 | **Not Applicable** |

| Get ItemsCount | **Method:** *getItemsCount()*

 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.getItemsCount();
 | **Not Applicable** |

| AddItem | **Method:** *addItem(text, content, index)*

 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.addItem("New item", "The accordion content",
2);
 | **Method:** *addItem(items, index)*

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.addItem([{ header: 'App',
content: 'text' }], 0)
 |

| Remove Item | **Method:** *removeItem(index)*

 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.removeItem(0);
 | **Method:**
removeItem(index)

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.removeItem(0)
 |

| Disable Items | **Property:** *DisabledItems*

@Html.EJ().Accordion("ejAccoordion").DisabledItems(new List<int>() { 0, 1 }).Render()
 | **Not
Applicable** |

| Enable Items | **Property:** *EnabledItems*

@Html.EJ().Accordion("ejAccoordion").EnabledItems(new List<int>() { 0, 1 }).Render()
 | **Not
Applicable** |

| Disable Item | **Method:** *disableItems([index])*

 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.disableItems([1]);
 | **Method:**
enableItem(index, false)

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.enableItem(0, false)

|

| Enable Item | **Method:** *enableItems([index])*

 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.enableItems([1]);
 | **Method:**
enableItem(index, true)

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.enableItem(0, true)

|

| Hide Item | **Not Applicable** | **Method:** *hideItem(index, true)*

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.hideItem(0, true)
 |

| SelectedItemIndex | **Property:** *SelectedItemIndex*

 @Html.EJ().Accordion("ejAccoordion").SelectedItemIndex("1").Render()
 | **Not Applicable** |

| Select | **Not Applicable** | **Method:** *select(index)*

 var accordion =
 document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.select(0)
 |

| BeforeActivate | **Event:** *BeforeActivate*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e=>e.BeforeActivate("beforeActivate")).Render()
)
 | **Event:** *Expanding*

 @Html.EJS().Accordion("ej2Accordion").Expanding("onExpanding").Render()
 |

| Activate | **Event:** *Activate*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e=>e.Activate("onActivate")).Render()
 |
Event: *Expanded*

 @Html.EJS().Accordion("ej2Accordion").Expanded("onExpanded").Render()
 |

| beforeInActivate | **Event:** *BeforeInactivate*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e=>e.BeforeInActivate("beforeInActivate")).Render()
 ()
 | **Not Applicable** |

| InActive | **Event:** *InActivate*

 @Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e=>e.InActive("inActive")).Render()
 | **Not Applicable** |

| Clicked | **Not Applicable** | **Event:** *Clicked*

 @Html.EJS().Accordion("ej2Accordion").Clicked("onClicked").Render()
 |

Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Collapsible | **Property:** *Collapsible*

 @Html.EJ().Accordion("ejAccoordion").Collapsible(true).Render()
 | **Not Applicable** |

| Collapse speed | **Property:** *CollapseSpeed*

 @Html.EJ().Accordion("ejAccoordion").CollapseSpeed("500").Render()
 | **Not Applicable** |

| Custom class | **Property:** *CssClass*

 @Html.EJ().Accordion("ejAccoordion").CssClass("custom").Render()
 | **Not Applicable** |

| CustomIcon class | **Property:** *CustomIcon*

 @Html.EJ().Accordion("ejAccoordion").CustomIcon(@ViewBag.CustomIcon).Render()
 | **Not Applicable** |

| Enabled | **Property:** *Enabled*

 @Html.EJ().Accordion("ejAccoordion").Enabled(false).Render()
 | **Not Applicable** |

| Events | **Property:** *Events*

 @Html.EJ().Accordion("ejAccoordion").Events("mouseover").Render()
 | **Not Applicable** |

| Expand speed | **Property:** *ExpandSpeed*

 @Html.EJ().Accordion("ejAccoordion").ExpandSpeed(300).Render()
 | **Not Applicable** |

| Height | **Property:** *Height*

 @Html.EJ().Accordion("ejAccoordion").Height("400px").Render()
 | **Property:** *Height*

 @Html.EJS().Accordion("ej2Accordion").Height("500px").Render()
 |

| HeightAdjustMode | **Property:** *HeightAdjustMode*

@Html.EJ().Accordion("ejAccoordion").HeightAdjustMode(HeightAdjustMode.Fill).Render()
 | **Not Applicable** |

| HtmlAttributes | **Property:** *HtmlAttributes*

@Html.EJ().Accordion("ejAccoordion").HtmlAttributes("").Render()
 | **Not Applicable** |

| MultipleOpen | **Property:** *EnableMultipleOpen*

@Html.EJ().Accordion("ejAccoordion").EnableMultipleOpen(true).Render()
 | **Property:** *ExpandMode*

@Html.EJS().Accordion("ej2Accordion").ExpandMode(Syncfusion.EJ2.Navigations.ExpandMode.Single).Render()
 |

| Persistence | **Property:** *EnablePersistence*

@Html.EJ().Accordion("ejAccoordion").EnablePersistence(true).Render()
 | **Property:** *EnablePersistence*

@Html.EJS().Accordion("ej2Accordion").EnablePersistence(true).Render()
 |

| ShowRounderCorner | **Property:** *ShowRoundedCorner*

/>@Html.EJ().Accordion("ejAccoordion").ShowRoundedCorner(true).Render()
 | **Not Applicable** |

| Width | **Property:** *Width*

 @Html.EJ().Accordion("ejAccoordion").Width("500px").Render()
 | **Property:** *Width*

 @Html.EJS().Accordion("ej2Accordion").Width("500px").Render()
 |

| Enable | **Method:** *enable()*

 var accordion = \$("#accordion").data("ejAccordion");

accordion.enable();
 | **Not Applicable** |

| Disable | **Method:** *disable()*

 var accordion = \$("#accordion").data("ejAccordion");

accordion.disable();
 | **Not Applicable** |

| Show | **Method:** *show()*

 var accordion = \$("#accordion").data("ejAccordion");

accordion.show();
 | **Not Applicable** |

| Hide | **Method:** *hide()*

 var accordion = \$("#accordion").data("ejAccordion");

accordion.hide();
 | **Not Applicable** |

| Destroy | **Method:** *destroy()*

 var accordion = \$("#accordion").data("ejAccordion");

accordion.destroy();
 | **Method:** *destroy()*

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.destroy();
 |

| Refresh | **Method:** *refresh()*

 var accordion = \$("#accordion").data("ejAccordion");

accordion.refresh();
 | **Method:** *refresh()*

 var accordion =
document.getElementById('ej2Accordion').ej2_instances[0];
 accordion.refresh();
 |

| Created | **Event:** *Create*

@Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e=>e.Create("onCreate")).Render()
 |
Event: *Created*

 @Html.EJS().Accordion("ej2Accordion").Created("onCreated").Render()

/> |

| Destroyed | **Event:** *Destroy*

@Html.EJ().Accordion("ejAccoordion").ClientSideEvents(e=>e.Destroy("onDestroy")).Render()
 |
Event: *Destroyed*

@Html.EJS().Accordion("ej2Accordion").Destroyed("onDestroyed").Render()
 |

Accumulation Chart

<!-- markdownlint-disable MD036 -->

Getting Started with ASP.NET MVC AccumulationChart Control

This section briefly explains about how to include [ASP.NET MVC AccumulationChart](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
```

```
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC AccumulationChart control

Now, add the Syncfusion ASP.NET MVC AccumulationChart control in `~/Home/Index.cshtml` page.

CSHTML

```
@(Html.EJS().AccumulationChart("container").Render())
```

Pie Series

By default, the pie series will be rendered when assigning the JSON data to the series using the [dataSource](#) property. Map the field names in the JSON data to the [xName](#) and [yName](#) properties of the series.

CSHTML

```
@model List<AccumulationChartSample.Controllers.PieChartData>
@(Html.EJS().AccumulationChart("container").Title("Mobile Browser
Statistics")
    .LegendSettings(ls => ls.Visible(false)).EnableSmartLabels(true)
    .Series(series =>
    {
        series.DataLabel(dl => dl.Visible(true).Name("text"))
            .XName("xValue")
            .YName("yValue")
            .Name("Browser")
            .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
            .DataSource(ViewBag.dataSource).Add();
    })
    .Render())
```

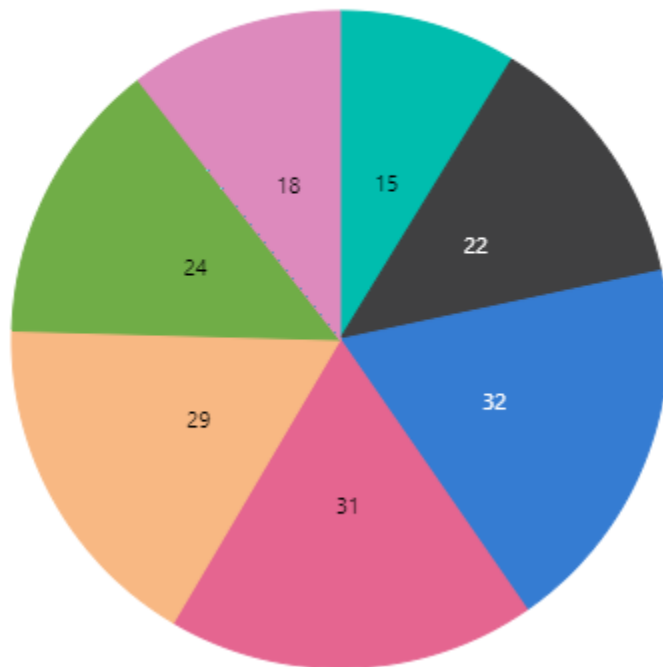
HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text = "37%",
            fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text = "17%",
            fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text = "19%",
            fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text = "4%",
            fill="#81e2a1"},
    };
    return View(chartData);
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC AccumulationChart control will be rendered in the default web browser.

Mobile Browser Statistics



Note: [View Sample in GitHub.](#)

Pie & Doughnut in ASP.NET MVC Accumulation Chart Component

Pie Chart

To render a pie series, use the series [Type](#) as **Pie**.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

PIE.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Radius Customization

By default, radius of the pie series will be 80% of the size (minimum of chart width and height). You can customize this using [Radius](#) property of the series.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
```

```

        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Radius("50%")
        .ExplodeIndex(0).Add();
    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false)).Render()

```

RADIUS.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Pie Center

The center position of the pie can be changed by Center X and Center Y. By default, center value of the pie series x and y is 50%. You can customize this using [Center](#) property of the series.

CSHTML

```

@{
    var piecenter = new
    {
        x = "60%",
        y = "60%"
    };
}

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataLabel(dl =>
    dl.Visible(true).Name("text").Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Inside).Font(ft => ft.FontWeight("600")))
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Radius("70%")
        .DataSource(ViewBag.dataSource).Add();
}

```

```

    })
    .Center(piecenter)
    .EnableSmartLabels(true)
    .EnableAnimation(false)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false))
    .Tooltip(tp => tp.Enable(true)).Render()

```

PIECENTER.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieData { xValue = "Chrome", yValue = 37, text = "37%" },
        new PieData { xValue = "UC Browser", yValue = 17, text = "17%" },
        new PieData { xValue = "iPhone", yValue = 19, text = "19%" },
        new PieData { xValue = "Others", yValue = 4, text = "4%" },
        new PieData { xValue = "Opera", yValue = 11, text = "11%" },
        new PieData { xValue = "Android", yValue = 12, text = "12%" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Various Radius Pie Chart

You can use radius mapping to render the slice with different [Radius](#) pie and also use [Border](#), fill properties to customize the point. dataLabel is used to represent individual data and its value.

CSHTML

```

@Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .XName("xValue")
        .YName("yValue")
        .DataLabel(dl =>
        dl.Visible(true).Name("xValue").Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Outside))
        .InnerRadius("20%")
        .Radius("r")
        .DataSource(ViewBag.dataSource).Add();
    })

```

```

    })
    .EnableSmartLabels(true)
    .EnableAnimation(true)
    .Tooltip(tp => tp.Enable(true).Format("${point.x} <br />Composition:
<b>${point.y}</b>"))
    .LegendSettings(leg => leg.Visible(true)).Render()

```

VARIOUS-RADIUS.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieRadiusChartData { xValue = "Argentina", yValue =
505370, r = "100"},
        new PieRadiusChartData { xValue = "Belgium", yValue =
551500, r = "118.7"},
        new PieRadiusChartData { xValue = "Cuba", yValue = 312685 ,
r = "124.6"},
        new PieRadiusChartData { xValue = "Dominican Republic",
yValue = 350000 , r = "137.5"},
        new PieRadiusChartData { xValue = "Egypt", yValue = 301000 ,
r = "150.8"},
        new PieRadiusChartData { xValue = "Kazakhstan", yValue =
300000, r = "155.5"},
        new PieRadiusChartData { xValue = "Somalia", yValue =
357022, r = "160.6"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string r;
}

```

Doughnut Chart

To achieve a doughnut in pie series, customize the [InnerRadius](#) property of the series. By setting value greater than 0%, a doughnut will appear. The [InnerRadius](#) property takes value from 0% to 100% of the pie radius.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .InnerRadius("40%")
        .ExplodeIndex(0).Add();
}

```

```

    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false)).Render()

```

DOUGHNUT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Start and End angles

You can customize the start and end angle of the pie series using the [StartAngle](#) and [EndAngle](#) properties. The default value of [StartAngle](#) is 0 degree, and [EndAngle](#) is 360 degrees. By customizing this, you can achieve a semi pie series.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .StartAngle(90)
        .EndAngle(270).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

START-ANGLE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },

```



```

        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Color & Text Mapping

The fill color and the text in the data source can be mapped to the chart using **PointColorMapping** in series and **Name** in datalabel respectively.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .PointColorMapping("fill")
        .DataLabel(ViewBag.datalabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

MAP.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text =
"17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData

```

```

{
    public string xValue;
    public double yValue;
}

```

Customization

Individual points can be customized using the **PointRender** event.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Add();
})
.EnableSmartLabels(true)
.PointRender("pointRender")
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

<script>
var pointRender = function (args) {
    if ((args.point.xValue as string).indexOf('iPhone') > -1) {
        args.fill = '#f4bc42';
    }
    else {
        args.fill = '#597cf9';
    }
};

</script>

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Hide pie or doughnut border

By default, the border will appear in the pie/doughnut chart while mouse hover on the chart. You can disable the the border by setting `EnableBorderOnMouseMove` property is `false`.

CSHTML

```
@Html.EJS().AccumulationChart("container").EnableBorderOnMouseMove(false).Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Add();
})

.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

BORDER.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Color Palette

You can customize the color the of the point using the `Palettes` property.

CSHTML

```
@Html.EJS().AccumulationChart("container").EnableBorderOnMouseMove(false).Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
```

```

        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .palettes("['teal', 'skyblue', 'green', 'red']")
        .Add();
    })

    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false)).Render()

```

PALETTE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Multi-level pie chart

You can achieve a multi-level drill down in pie and doughnut charts using [PointClick](#) event. If user clicks any point in the chart, that corresponding data will be shown in the next level and so on based on point clicked.

You can also achieve drill-up (back to the initial state) by using [ChartMouseClicked](#) event. In below sample, you can drill-up by clicking back button in the center of the chart

CSHTML

```

<div>
    @(Html.EJS().AccumulationChart("Chart").Series(
        series =>
        {
            series.DataLabel(dl =>
                dl.Visible(true).Name("text").Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Inside).Font(ft => ft.FontWeight("600"))).
                XName("x").
                YName("y").
                Name("Project").
                Type(Syncfusion.EJ2.Charts.AccumulationType.Pie).
                Radius("70%").
                StartAngle(0).
                EndAngle(360).
                DataSource(ViewBag.dataSource).
            )
        }
    ))

```

```

        Explode(false).Add();
    }).EnableSmartLabels(true).Title("Automobile Sales by
Category").TextRender("textRender").LegendSettings(
    legend =>
    {
        legend.Visible(false);
    })
    .PointClick("pointClick").ChartMouseClick("chartMouseClick").Render()
)

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.dataSource)
.Columns(col =>
{
    col.Field("x").HeaderText("Vehicles").Type("string").Add();
    col.Field("y").HeaderText("Sales").Type("string").Add();
}).Render()
</div>
<script>
    var pointIndex = -1;
    var data = [
        {
            x: 'SUV',
            y: 25,
            z: [
                {
                    title: 'Automobile Sales in the SUV Segment',
                    x: 'Toyota',
                    y: 8,
                    z: [
                        { x: '2000', y: 20 },
                        { x: '2001', y: 30 },
                        { x: '2002', y: 40 },
                    ],
                },
                { x: 'Ford', y: 12 },
                { x: 'GM', y: 17 },
                { x: 'Renault', y: 6 },
            ],
        },
        {
            x: 'Car',
            y: 37,
            z: [
                { title: 'Automobile Sales in the Car Segment', x: 'Toyota',
                    y: 7 },
                { x: 'Chrysler', y: 12 },
                { x: 'Nissan', y: 9 },
                { x: 'Ford', y: 15 },
            ],
        },
        {
            x: 'Pickup',
            y: 15,
            z: [
                { title: 'Automobile Sales in the Pickup Segment', x:
                    'Nissan', y: 9 },
                { x: 'Chrysler', y: 4 },
            ],
        },
    ],
    pointIndex = 0;
    var chart = new AccumulationChart({
        data: data,
        pointIndex: pointIndex,
        chartMouseClick: "chartMouseClick",
        pointClick: "pointClick",
        textRender: "textRender",
        legend: "legend",
        title: "Automobile Sales by Category",
        smartLabels: true,
        explode: false,
        legendSettings: {
            visible: false
        }
    });
    chart.render();
</script>

```

```

        { x: 'Ford', y: 7 },
        { x: 'Toyota', y: 20 },
    ],
},
{
    x: 'Minivan',
    y: 23,
    z: [
        { title: 'Automobile Sales in the Minivan Segment', x:
'Hummer', y: 11 },
        { x: 'Ford', y: 5 },
        { x: 'GM', y: 12 },
        { x: 'Chrysler', y: 3 },
    ],
},
};
var textRender = function (args) {
    args.text = args.point.x + ' ' + args.point.y + ' %';
};
var chartMouseClicked = function (args) {
    var pie = document.getElementById("Chart").ej2_instances[0];
    var grid = document.getElementById("Grid").ej2_instances[0];
    if (args.target.indexOf('back') > -1) {
        if (pie.series[0].name === 'Level 3') {
            pie.series[0].dataSource = data[window.pointIndex].z;
            pie.series[0].name = 'Level 2';
            pie.title = data[window.pointIndex].z[0].title;
            pie.series[0].innerRadius = '30%';
            grid.dataSource = pie.series[0].dataSource;
            grid.columns[0].headerText = data[window.pointIndex].x;
            grid.refresh();
            pie.refresh();
        } else if (pie.series[0].name === 'Level 2') {
            pie.series[0].dataSource = data;
            pie.series[0].name = 'Level 1';
            pie.series[0].innerRadius = '0%';
            pie.title = 'Automobile Sales by Category';
            pie.annotations = [{}];
            pie.pointClick = pointClick;
            grid.dataSource = pie.series[0].dataSource;
            grid.columns[0].headerText = 'Vehicle';
            grid.refresh();
            pie.refresh();
        }
    }
    grid.dataSource = pie.series[0].dataSource;
};
var pointClick = function (args) {
    var pie = document.getElementById("Chart").ej2_instances[0];
    var grid = document.getElementById("Grid").ej2_instances[0];
    if (
        ej.charts.getElement(
            pie.element.id +
            '_Series_' +
            args.seriesIndex +
            '_Point_' +
            args.pointIndex

```

```

    )
    {
        pie.series[0].dataSource = data[args.pointIndex].z;
        pie.title = data[args.pointIndex].z[0].title;
        window.pointIndex = args.pointIndex;
        pie.series[0].name = 'Level 2';
        pie.series[0].innerRadius = '30%';
        pie.annotations = [
            {
                content:
                    '<div id="back"
style="cursor:pointer;padding:3px;width:30px; height:30px;">' +
                    '',
                region: 'Series',
                x: '50%',
                y: '50%',
            },
        ];
        pie.pointClick = click;
    }
    grid.dataSource = pie.series[0].dataSource;
    grid.columns[0].headerText = data[args.pointIndex].x;
    grid.refresh();
    pie.refresh();
}
var click = function (args) {
    var pie = document.getElementById("Chart").ej2_instances[0];
    var grid = document.getElementById("Grid").ej2_instances[0];
    if (pie.series[0].name !== 'Level 3') {
        switch (args.pointIndex) {
            case 0:
                pie.series[0].dataSource = data[0].z[0].z;
                pie.title = 'SUV Sales by Years';
                pie.series[0].name = 'Level 3';
                grid.columns[0].headerText = 'Year';
                grid.refresh();
                pie.refresh();
                break;
        }
        grid.dataSource = pie.series[0].dataSource;
    }
};
</script>

```

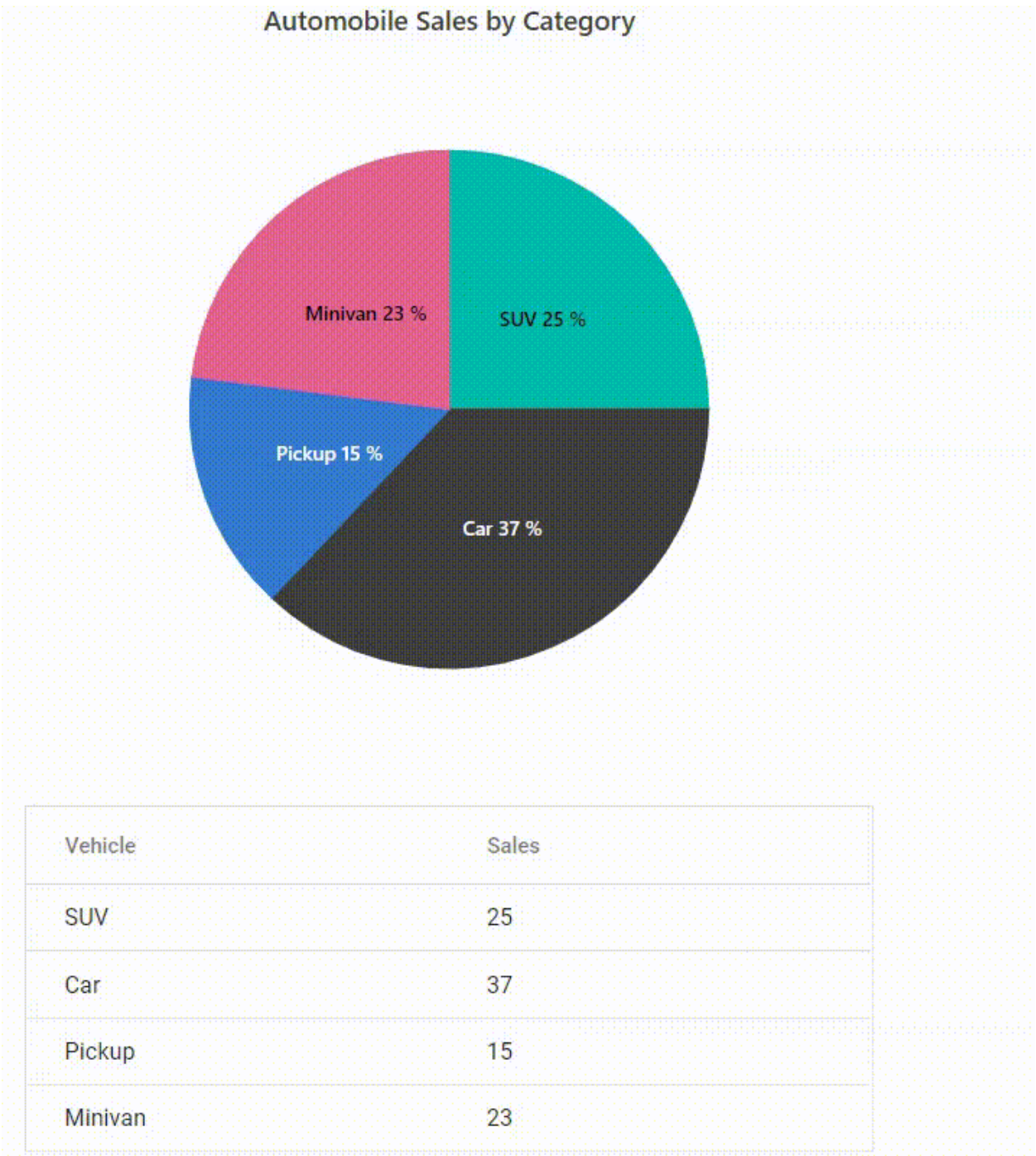
DEFAULT.CS

```

public IActionResult Index()
{
    List<CategoryData> chartData = new List<CategoryData>
    {
        new CategoryData { x = "SUV", y = 25 },
        new CategoryData { x = "Car", y = 37 },
        new CategoryData { x = "Pickup", y = 15 },
        new CategoryData { x = "Minivan", y = 23 },
    }
}

```

```
};  
ViewBag.datalabel = new  
{  
    visible = true,  
    position = "Inside",  
    name = "text",  
    font = new  
    {  
        fontWeight = "600"  
    }  
};  
ViewBag.dataSource = chartData;  
return View();  
}  
public class CategoryData  
{  
    public string x;  
    public double y;  
}
```

See Also

- [Data label](#)
- [Grouping](#)

Pyramid Chart

To render a pyramid series, use the series [Type](#) as `Pyramid`.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pyramid).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

DEFAULT.CS

```
public ActionResult Index()
{
    List<PyramidChartData> chartData = new List<PyramidChartData>
    {
        new PyramidChartData { xValue = "Chrome", yValue = 37 },
        new PyramidChartData { xValue = "UC Browser", yValue = 17 },
        new PyramidChartData { xValue = "iPhone", yValue = 19 },
        new PyramidChartData { xValue = "Others", yValue = 14 },
        new PyramidChartData { xValue = "Opera", yValue = 11 },
        new PyramidChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PyramidChartData
{
    public string xValue;
    public double yValue;
}
```

Mode

The Pyramid chart supports linear and surface modes of rendering. The default type of the **PyramidMode** is **Linear**.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .PyramidMode("Surface")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pyramid).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

MODE.CS

```

public ActionResult Index()
{
    List<PyramidChartData> chartData = new List<PyramidChartData>
    {
        new PyramidChartData { xValue = "Chrome", yValue = 37 },
        new PyramidChartData { xValue = "UC Browser", yValue = 17 },
        new PyramidChartData { xValue = "iPhone", yValue = 19 },
        new PyramidChartData { xValue = "Others", yValue = 4 },
        new PyramidChartData { xValue = "Opera", yValue = 11 },
        new PyramidChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PyramidChartData
{
    public string xValue;
    public double yValue;
}

```

Size

The size of the pyramid chart can be customized by using the **Width** and **Height** properties.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Width("60%")
        .Height("80%")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pyramid).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

SIZE.CS

```

public ActionResult Index()
{
    List<PyramidChartData> chartData = new List<PyramidChartData>
    {
        new PyramidChartData { xValue = "Chrome", yValue = 37 },
        new PyramidChartData { xValue = "UC Browser", yValue = 17 },
        new PyramidChartData { xValue = "iPhone", yValue = 19 },
        new PyramidChartData { xValue = "Others", yValue = 4 },
    }
}

```

```

        new PyramidChartData { xValue = "Opera", yValue = 11 },
        new PyramidChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PyramidChartData
{
    public string xValue;
    public double yValue;
}

```

Gap Between the Segments

Pyramid chart provides options to customize the space between the segments by using the **GapRatio** property of the series. It ranges from 0 to 1.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .GapRatio(0.2)

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pyramid).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

GAP.CS

```

public ActionResult Index()
{
    List<PyramidChartData> chartData = new List<PyramidChartData>
    {
        new PyramidChartData { xValue = "Chrome", yValue = 37 },
        new PyramidChartData { xValue = "UC Browser", yValue = 17 },
        new PyramidChartData { xValue = "iPhone", yValue = 19 },
        new PyramidChartData { xValue = "Others", yValue = 4 },
        new PyramidChartData { xValue = "Opera", yValue = 11 },
        new PyramidChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PyramidChartData
{
    public string xValue;
    public double yValue;
}

```

Explode

Points can be exploded on mouse click by setting the **Explode** property to true. You can also explode the point on load using **ExplodeIndex**. Explode distance can be set by using **ExplodeOffset** property.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Explode(true)
        .ExplodeOffset("10%")
        .ExplodeIndex(0)
        .ExplodeAll(false)

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pyramid).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

EXPLODE.CS

```
public ActionResult Index()
{
    List<PyramidChartData> chartData = new List<PyramidChartData>
    {
        new PyramidChartData { xValue = "Chrome", yValue = 37 },
        new PyramidChartData { xValue = "UC Browser", yValue = 17 },
        new PyramidChartData { xValue = "iPhone", yValue = 19 },
        new PyramidChartData { xValue = "Others", yValue = 4 },
        new PyramidChartData { xValue = "Opera", yValue = 11 },
        new PyramidChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PyramidChartData
{
    public string xValue;
    public double yValue;
}
```

Customization

Individual points can be customized using the **PointRender** event.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
```

```

        .YName("yValue")
        .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pyramid).Add();
    })
    .EnableSmartLabels(true)
    .PointRender("pointRender")
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false)).Render()

<script>
    var pointRender = function (args) {
        if ((args.point.xValue as string).indexOf('iPhone') > -1) {
            args.fill = '#f4bc42';
        }
        else {
            args.fill = '#597cf9';
        }
    };
</script>

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<PyramidChartData> chartData = new List<PyramidChartData>
    {
        new PyramidChartData { xValue = "Chrome", yValue = 37 },
        new PyramidChartData { xValue = "UC Browser", yValue = 17 },
        new PyramidChartData { xValue = "iPhone", yValue = 19 },
        new PyramidChartData { xValue = "Others", yValue = 4 },
        new PyramidChartData { xValue = "Opera", yValue = 11 },
        new PyramidChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PyramidChartData
{
    public string xValue;
    public double yValue;
}

```

See Also

- [Data label](#)
- [Grouping](#)

Funnel Chart

To render a funnel series, use the series [Type](#) as **Funnel**.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
```

```

        {
            series.DataSource(ViewBag.dataSource)
                .XName("xValue")
                .YName("yValue")
                .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel).Add();
        })
        .EnableSmartLabels(true)
        .Title("Mobile Browser Statistics")
        .LegendSettings(ls => ls.Visible(false)).Render()
    )

```

DEFAULT.CS

```

public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelChartData { xValue = "Chrome", yValue = 37 },
        new FunnelChartData { xValue = "UC Browser", yValue = 17 },
        new FunnelChartData { xValue = "iPhone", yValue = 19 },
        new FunnelChartData { xValue = "Others", yValue = 4 },
        new FunnelChartData { xValue = "Opera", yValue = 11 },
        new FunnelChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class FunnelChartData
{
    public string xValue;
    public double yValue;
}

```

Size

The size of the funnel chart can be customized by using the **Width** and **Height** properties.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Width("60%")
        .Height("80%")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

SIZE.CS

```

public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelChartData { xValue = "Chrome", yValue = 37 },
        new FunnelChartData { xValue = "UC Browser", yValue = 17 },
        new FunnelChartData { xValue = "iPhone", yValue = 19 },
        new FunnelChartData { xValue = "Others", yValue = 4 },
        new FunnelChartData { xValue = "Opera", yValue = 11 },
        new FunnelChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class FunnelChartData
{
    public string xValue;
    public double yValue;
}

```

Neck Size

The funnel's neck size can be customized by using the `NeckWidth` and `NeckHeight` properties.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .NeckWidth("15%")
        .NeckHeight("18%")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

NECK-SIZE.CS

```

public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelChartData { xValue = "Chrome", yValue = 37 },
        new FunnelChartData { xValue = "UC Browser", yValue = 17 },
        new FunnelChartData { xValue = "iPhone", yValue = 19 },
        new FunnelChartData { xValue = "Others", yValue = 4 },
        new FunnelChartData { xValue = "Opera", yValue = 11 },
        new FunnelChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
}

```



```

        return View();
    }
    public class FunnelChartData
    {
        public string xValue;
        public double yValue;
    }

```

Gap Between the Segments

Funnel chart provides options to customize the space between the segments by using the **GapRatio** property of the series. It ranges from 0 to 1.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .GapRatio(0.08)

.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

GAP.CS

```

public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelChartData { xValue = "Chrome", yValue = 37 },
        new FunnelChartData { xValue = "UC Browser", yValue = 17 },
        new FunnelChartData { xValue = "iPhone", yValue = 19 },
        new FunnelChartData { xValue = "Others", yValue = 4 },
        new FunnelChartData { xValue = "Opera", yValue = 11 },
        new FunnelChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class FunnelChartData
{
    public string xValue;
    public double yValue;
}

```

Explode

Points can be exploded on mouse click by setting the **Explode** property to true. You can also explode the point on load using **ExplodeIndex**. Explode distance can be set by using **ExplodeOffset** property.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Explode(true)
        .ExplodeOffset("10%")
        .ExplodeIndex(0)

.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

EXPLODE.CS

```
public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelChartData { xValue = "Chrome", yValue = 37 },
        new FunnelChartData { xValue = "UC Browser", yValue = 17 },
        new FunnelChartData { xValue = "iPhone", yValue = 19 },
        new FunnelChartData { xValue = "Others", yValue = 4 },
        new FunnelChartData { xValue = "Opera", yValue = 11 },
        new FunnelChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class FunnelChartData
{
    public string xValue;
    public double yValue;
}
```

Smart Data Label

It provides the data label smart arrangement of the funnel and pyramid series. The overlap data label will be placed on left side of the funnel/pyramid series.

CSHTML

```
@(Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {

sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel)
        .XName("xValue")
        .YName("yValue")
        .DataLabel(dl =>
dl.Visible(true).Name("text").Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Inside).Font(ft => ft.FontWeight("600")))
    })
    .Render()
```

```

        .Width("60%")
        .Height("80%")
        .NeckWidth("15%")
        .GapRatio(0.03)
        .NeckHeight("18%")
        .Explode(true)
        .DataSource(ViewBag.dataSource).Add();
    })
    .EnableSmartLabels(true)
    .Tooltip(tp => tp.Enable(true).Format("${point.x}
Composition: <b>${point.y}%</b>"))
    .LegendSettings(leg => leg.Visible(false))
    .Load("load").Resized("resize")
    .Title("Top population countries in the world
2017").Render()

```

DATA-LABEL.CS

```

public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelData { xValue = "China", yValue = 1409517397, text
= "China" },
        new FunnelData { xValue = "India", yValue = 1339180127, text
= "India" },
        new FunnelData { xValue = "United States", yValue =
324459463, text = "United States" },
        new FunnelData { xValue = "Indonesia", yValue = 263991379,
text = "Indonesia" },
        new FunnelData { xValue = "Brazil", yValue = 209288278, text
= "Brazil" },
        new FunnelData { xValue = "Pakistan", yValue = 197015955,
text = "Pakistan" },
        new FunnelData { xValue = "Nigeria", yValue = 190886311,
text = "Nigeria" },
        new FunnelData { xValue = "Bangladesh", yValue = 164669751,
text = "Bangladesh" },
        new FunnelData { xValue = "Russia", yValue = 143989754, text
= "Russia" },
        new FunnelData { xValue = "Mexico", yValue = 129163276, text
= "Mexico" },
        new FunnelData { xValue = "Japan", yValue = 127484450, text
= " Japan" },
        new FunnelData { xValue = "Ethiopia", yValue = 104957438,
text = "Ethiopia" },
        new FunnelData { xValue = "Philippines", yValue = 104918090,
text = "Philippines" },
        new FunnelData { xValue = "Egypt", yValue = 97553151, text =
"Egypt" },
        new FunnelData { xValue = "Vietnam", yValue = 95540800, text
= "Vietnam" },
        new FunnelData { xValue = "Germany", yValue = 82114224, text
= "Germany" }
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class FunnelChartData
    {
        public string xValue;
        public double yValue;
        public string text;
    }

```

Customization

Individual points can be customized using the **PointRender** event.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Funnel).Add();
})
.EnableSmartLabels(true)
.PointRender("pointRender")
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

<script>
var pointRender = function (args) {
    if ((args.point.xValue as string).indexOf('iPhone') > -1) {
        args.fill = '#f4bc42';
    }
    else {
        args.fill = '#597cf9';
    }
};

</script>

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<FunnelChartData> chartData = new List<FunnelChartData>
    {
        new FunnelChartData { xValue = "Chrome", yValue = 37 },
        new FunnelChartData { xValue = "UC Browser", yValue = 17 },
        new FunnelChartData { xValue = "iPhone", yValue = 19 },
        new FunnelChartData { xValue = "Others", yValue = 4 },
        new FunnelChartData { xValue = "Opera", yValue = 11 },
        new FunnelChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class FunnelChartData

```

```
{
    public string xValue;
    public double yValue;
}
```

See Also

- [Data label](#)
- [Grouping](#)

Data Label in ASP.NET MVC Accumulation Chart Component

Data label can be added to a chart series by enabling the [Visible](#) option in the dataLabel property.

CSHTML

```
Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.dataLabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

DEFAULT.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text
= "17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}
```

Positioning

Accumulation chart provides support for placing the data label either **Inside** or **Outside** the chart.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()
```

POSITION.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text =
"17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}
```

Smart labels

Data labels will be arranged smartly without overlapping with each other. You can enable or disable this feature using the [EnableSmartLabels](#) property.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
```

```

        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel).Add();
    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false)).Render()

```

SMARTLABELS.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text =
"17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Data Label Template

Label content can be formatted by using the template option. Inside the template, you can add the placeholder text `${point.x}` and `${point.y}` to display corresponding data points x & y value. Using [Template](#) property, you can set data label template in chart.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

TEMPLATE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text
= "17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4 , text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Connector Line

Connector line will be visible when the data label is placed **Outside** the chart. The connector line can be customized using the **Type**, **Color**, **Width**, **Length** and **DashArray** properties.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

CONNECTOR.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},

```



```

        new PieChartData { xValue = "UC Browser", yValue = 17, text =
= "17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Text Mapping

Text from the data source can be mapped to data label using **Name** property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.dataLabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

MAP.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text =
"17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData

```

```

{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Format

Data label for the accumulation chart can be formatted using [Format](#) property. You can use the global formatting options, such as 'n', 'p', and 'c'.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(dl=>dl.Visible(true).Format("n2")).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

FORMAT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text
= "17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Value	Format	Resultant Value	Description
-------	--------	-----------------	-------------

1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Customization

Individual text can be customized using the **TextRender** event.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.TextRender("textRender")
.LegendSettings(ls => ls.Visible(false)).Render()

<script>
var textRender = function (args) {
    if (args.text.indexOf('iPhone') > -1) {
        args.color = 'red';
        args.border.width = 1;
    }
};

</script>
```

CUSTOM.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text
= "17%", fill="#ffa060"},
    };
}
```

```

        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Text wrap

When the data label text exceeds the container, the text can be wrapped by using ['textWrap'](#) property. End user can also wrap the data label text based on ['maxWidth'](#) property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .StartAngle(270)
        .endAngle(90)
        .TooltipMappingName('tooltipMappingName')
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)

    .DataLabel(dl=>dl.Visible(true).Position('Inside').MaxWidth(100).TextWrap('Width').Name('text').EnableRotation(true)).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(false)).Render()

```

TEXTWRAP.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 100, text= "Chrome
(100M)<br>40%", tooltipMappingName= "40%"},
        new PieChartData { x= "UC Browser", y= 40, text= "UC
Browser (40M)<br>16%", tooltipMappingName= "16%"},
        new PieChartData { x= "Opera", y= 30, text= "Opera
(30M)<br>12%", tooltipMappingName= "12%"},
        new PieChartData { x= "Safari", y= 30, text= "Safari
(30M)<br>12%", tooltipMappingName= "12%"},
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class PieChartData
    {
        public string x;
        public double y;
        public string text;
        public string tooltipMappingName;
    }

```

Show percentages in data labels of pie chart

You can show the percentages in data labels of pie chart using `textRender` event and `template` option.

Using textRender event

You can customize the data label of pie chart using [TextRender](#) event as follows to show percentage.

CSHTML

```

Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.dataSource).Add();
})
.EnableSmartLabels(true)
.TextRender("textRender")
.LegendSettings(ls => ls.Visible(false)).Render()
<script>
var textRender = function (args) {
    args.text = args.point.percentage + "%";
};
</script>

```

PERCENTAGE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text = "37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text = "17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text = "19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text = "4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData

```

```

{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```

Using template

You can display the percentage values in data label of pie chart using **template** option.

CSHTML

```

@ (Html.EJS().AccumulationChart("container")
    .LegendSettings(ls => ls.Visible(false)).EnableSmartLabels(true).
    Series(series =>
    {
        series.DataLabel(dl =>
        dl.Visible(true).Name("text").Template("<div
id='dataLabelTemplate'>${point.percentage}%</div>"))
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataSource(ViewBag.dataSource).Add());
    }).Render())

```

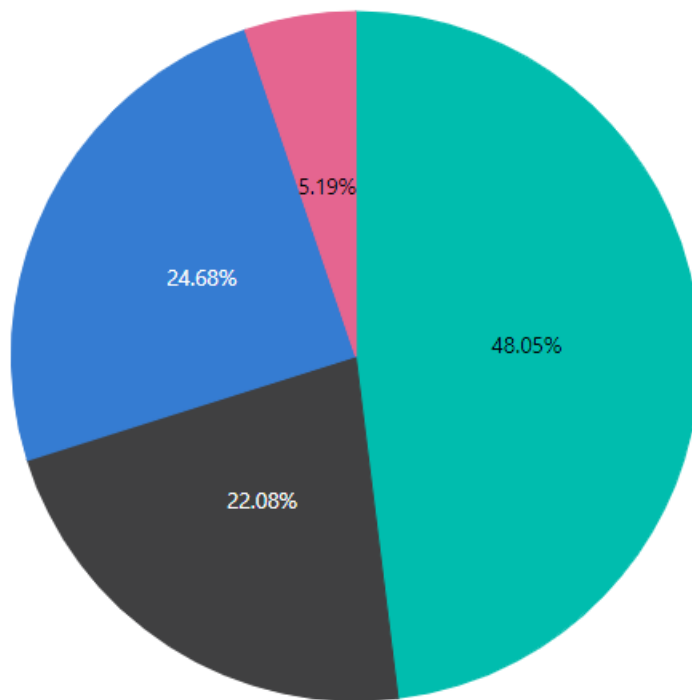
TEMPLATE-PERCENTAGE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37, text =
"37%", fill="#498fff"},
        new PieChartData { xValue = "UC Browser", yValue = 17, text
= "17%", fill="#ffa060"},
        new PieChartData { xValue = "iPhone", yValue = 19, text =
"19%", fill="#ff68b6"},
        new PieChartData { xValue = "Others", yValue = 4, text =
"4%", fill="#81e2a1"},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
    public string fill;
}

```



<!-- markdownlint-disable MD036 -->

Grouping

You can club/group few points of the series based on [GroupTo](#) property. For example, if the club value is 11, then the points with value less than 11 is grouped together and will be showed as a single point with label **others**. The property also takes value in percentage (percentage of total data points value).

CSHTML

```
@Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .XName("xValue")
        .YName("yValue")
        .Name("RIO")
        .Animation(ViewBag.animation)
        .DataSource(ViewBag.dataSource)
        .GroupTo("10").Add();
    })
    .EnableSmartLabels(true)
    .LegendSettings(leg => leg.Visible(false))
    .Title("RIO Olympics Gold").Render()
```

GROUP.CS

```
public IActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
```

```

        new GroupingChartData { xValue = "China", yValue = 26, text
= "China: 26" },
        new GroupingChartData { xValue = "Russia", yValue = 19, text
= "Russia: 19" },
        new GroupingChartData { xValue = "Germany", yValue = 17,
text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan", yValue = 12, text
= "Japan: 12" },
        new GroupingChartData { xValue = "France", yValue = 10, text
= "France: 10" },
        new GroupingChartData { xValue = "South Korea", yValue = 9,
text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain", yValue =
27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy", yValue = 8, text
= "Italy: 8" },
        new GroupingChartData { xValue = "Australia", yValue = 8,
text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands", yValue = 8,
text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary", yValue = 8,
text = "Hungary: 8" },
        new GroupingChartData { xValue = "Brazil", yValue = 7,
text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain", yValue = 7,
text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya", yValue = 6,
text = "Kenya: 6" }
    };
    ViewBag.dataSource = chartData;

    return View();
}
public class GroupingChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Pie Grouping

Broken Slice

You can visualize all points available in club/group points by clicking on the grouped point. For example, if 5 points are grouped together it will be showed as a single slice with label **others**. If we click on **others** slice it will explode and broke into 5 seperate slices.

CSHTML

```

@ (Html.EJS().AccumulationChart("container")
    .Series(sr =>
        {
            sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
            .XName("xValue")
            .YName("yValue")
            .Name("RIO")
        }
    )
)

```



```

        .Explode(true)
        .DataLabel(dl =>
dl.Visible(true).Name("text").Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Outside).ConnectorStyle(cs=>cs.Type(Syncfusion.EJ2.Charts.ConnectorType.Line).Length("5%")).Font(ft => ft.Size("14px"))
        .Animation(animate => animate.Enable(false))
        .Radius("70%")
        .StartAngle(0)
        .EndAngle(360)
        .InnerRadius("0%")
        .GroupTo("9")
        .GroupMode(Syncfusion.EJ2.Charts.GroupModes.Point)
        .DataSource(ViewBag.dataSource).Add();
    })
    .EnableSmartLabels(true)
    .Tooltip(tp => tp.Enable(false))
    .LegendSettings(leg => leg.Visible(false))
    .Title("RIO Olympics Gold").Render()
)

```

SLICE.CS

```

public ActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
        new GroupingChartData { xValue = "China", yValue = 26, text = "China: 26" },
        new GroupingChartData { xValue = "Russia", yValue = 19, text = "Russia: 19" },
        new GroupingChartData { xValue = "Germany", yValue = 17, text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan", yValue = 12, text = "Japan: 12" },
        new GroupingChartData { xValue = "France", yValue = 10, text = "France: 10" },
        new GroupingChartData { xValue = "South Korea", yValue = 9, text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain", yValue = 27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy", yValue = 8, text = "Italy: 8" },
        new GroupingChartData { xValue = "Australia", yValue = 8, text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands", yValue = 8, text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary", yValue = 8, text = "Hungary: 8" },
        new GroupingChartData { xValue = "Brazil", yValue = 7, text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain", yValue = 7, text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya", yValue = 6, text = "Kenya: 6" }
    };
    ViewBag.animation = new ChartAnimation { Enable = true };
}

```

```

    ViewBag.dataSource = chartData;
    return View();
}
public class GroupingChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Group Mode

Slice can also be grouped based on number of points by specifying the [GroupMode](#) to Point. For example, if the group to value is 11, accumulation chart will show 1st 11 points and will group remaining entries in the collection as a single point.

CSHTML

```

@(Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .XName("xValue")
        .YName("yValue")
        .Name("RIO")
        .Explode(true)
        .DataLabel(dl =>
        dl.Visible(true).Name("text").Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Outside).ConnectorStyle(cs=>cs.Type(Syncfusion.EJ2.Charts.ConnectorType.Line).Length("5%")).Font(ft => ft.Size("14px")))
        .Animation(animate => animate.Enable(false))
        .Radius("70%")
        .StartAngle(0)
        .EndAngle(360)
        .InnerRadius("0%")
        .GroupTo("3")
        .GroupMode(Syncfusion.EJ2.Charts.GroupModes.Point)
        .DataSource(ViewBag.dataSource).Add();
    })
    .EnableSmartLabels(true)
    .Tooltip(tp => tp.Enable(false))
    .LegendSettings(leg => leg.Visible(false))
    .Title("RIO Olympics Gold").Render()
)

```

GROUPMODE.CS

```

public ActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
        new GroupingChartData { xValue = "China", yValue =
26, text = "China: 26" },
        new GroupingChartData { xValue = "Russia", yValue =
19, text = "Russia: 19" },
    }
}

```

```

        new GroupingChartData { xValue = "Germany", yValue =
17, text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan", yValue =
12, text = "Japan: 12" },
        new GroupingChartData { xValue = "France", yValue =
10, text = "France: 10" },
        new GroupingChartData { xValue = "South Korea", yValue =
9, text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain", yValue =
27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy", yValue =
8, text = "Italy: 8" },
        new GroupingChartData { xValue = "Australia", yValue =
8, text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands", yValue =
8, text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary", yValue =
8, text = "Hungary: 8" },
        new GroupingChartData { xValue = "Brazil", yValue =
7, text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain", yValue =
7, text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya", yValue =
6, text = "Kenya: 6" }
    };
    ViewBag.animation = new ChartAnimation { Enable = true };
    ViewBag.dataSource = chartData;
    return View();
}

public class GroupingChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Customization

You can customize the grouped point and its data label using **PointRender** and **TextRender** event.

CSHTML

```

@Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
            .XName("xValue")
            .YName("yValue")
            .Name("RIO")
            .Animation(ViewBag.animation)
            .DataSource(ViewBag.dataSource)
            .GroupTo("10")
            .DataLabel(ViewBag.dataLabel).Add();
    })
    .EnableSmartLabels(true)
    .LegendSettings(leg => leg.Visible(false))
    .PointRender("pointRender").TextRender("textRender")

```

```

        .Title("RIO Olympics Gold").Render()
<script>
    var pointRender = function(args) {
        if ((args.point.x as string).indexOf('Others') > -1) {
            args.fill = '#D3D3D3';
        }
    }
    var textRender = function(args) {
        if (args.text.indexOf('Others') > -1) {
            args.text = 'Grouped Slices';
            args.color = 'red';
            args.border.width = 1;
        }
    }
</script>

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
        new GroupingChartData { xValue = "China", yValue = 26, text
= "China: 26" },
        new GroupingChartData { xValue = "Russia", yValue = 19, text
= "Russia: 19" },
        new GroupingChartData { xValue = "Germany", yValue = 17,
text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan", yValue = 12, text
= "Japan: 12" },
        new GroupingChartData { xValue = "France", yValue = 10, text
= "France: 10" },
        new GroupingChartData { xValue = "South Korea", yValue = 9,
text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain", yValue =
27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy", yValue = 8, text
= "Italy: 8" },
        new GroupingChartData { xValue = "Australia", yValue = 8,
text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands", yValue = 8,
text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary", yValue = 8,
text = "Hungary: 8" },
        new GroupingChartData { xValue = "Brazil", yValue = 7,
text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain", yValue = 7,
text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya", yValue = 6,
text = "Kenya: 6" }
    };
    ViewBag.dataSource = chartData;

    return View();
}
public class GroupingChartData

```

```

{
    public string xValue;
    public double yValue;
    public string text;
}

```

Empty Points

The data points those uses the **null** or **undefined** as value are considered as empty points. The empty data points are ignored and not plotted in the chart. You can customize those points, using the **EmptyPointSettings** property in series. The default mode of the empty point is **Gap**. Other supported modes are **Average** and **Zero**.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Profit")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel)
        .EmptyPointSettings(ViewBag.emptypoint).Add();
})
.EnableSmartLabels(true)
.Title("Annual Product-Wise Profit Analysis")
.LegendSettings(ls => ls.Visible(false))
.Load("load").Render()

```

EMPTY.CS

```

public ActionResult Index()
{
    List<EmptyPointsChartData> chartData = new
List<EmptyPointsChartData>
    {
        new EmptyPointsChartData { xValue = "Rice", yValue = 80
    },
        new EmptyPointsChartData { xValue = "Wheat", yValue = null
    },
        new EmptyPointsChartData { xValue = "Oil", yValue = 70
    },
        new EmptyPointsChartData { xValue = "Corn", yValue = 60
    },
        new EmptyPointsChartData { xValue = "Gram", yValue = null
    },
        new EmptyPointsChartData { xValue = "Milk", yValue = 70
    },
        new EmptyPointsChartData { xValue = "Peas", yValue = 80
    },
        new EmptyPointsChartData { xValue = "Fruit", yValue = 60
    },
        new EmptyPointsChartData { xValue = "Butter", yValue = null
    },
    };
}

```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class EmptyPointsChartData
    {
        public string xValue;
        public Nullable<double> yValue;
    }

```

Customization

Specific color for an empty point can be set by using the **Fill** property in **EmptyPointSettings** and the border for an empty point can be set by using the **Border** property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Profit")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataLabel(ViewBag.datalabel)
        .EmptyPointSettings(ViewBag.emptypoint).Add();
})
.EnableSmartLabels(true)
.Title("Annual Product-Wise Profit Analysis")
.LegendSettings(ls => ls.Visible(false))
.Load("load").Render()

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<EmptyPointsChartData> chartData = new
List<EmptyPointsChartData>
    {
        new EmptyPointsChartData { xValue = "Rice", yValue = 80
    },
        new EmptyPointsChartData { xValue = "Wheat", yValue = null
    },
        new EmptyPointsChartData { xValue = "Oil", yValue = 70
    },
        new EmptyPointsChartData { xValue = "Corn", yValue = 60
    },
        new EmptyPointsChartData { xValue = "Gram", yValue = null
    },
        new EmptyPointsChartData { xValue = "Milk", yValue = 70
    },
        new EmptyPointsChartData { xValue = "Peas", yValue = 80
    },
        new EmptyPointsChartData { xValue = "Fruit", yValue = 60
    },
    }
}

```

```

        new EmptyPointsChartData { xValue = "Butter", yValue = null
    },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class EmptyPointsChartData
{
    public string xValue;
    public Nullable<double> yValue;
}

```

Annotation

The annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

<!-- markdownlint-disable MD033 -->

By using the `<code>content</code>`

 option of annotation property, you can specify the Id of the element that needs to be displayed in the chart area.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.Annotations(an => {an
    .X("iPhone")
    .Y("19")
    .CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point)
    .Regions(Syncfusion.EJ2.Charts.Regions.Series)
    .Content(ViewBag.content).Add();
}).
.LegendSettings(ls => ls.Visible(false)).Render()

```

ANNOTATION.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 37 },
        new PieChartData { x = "UC Browser", y = 17 },
        new PieChartData { x = "iPhone", y = 19 },
        new PieChartData { x = "Others", y = 4 },
        new PieChartData { x = "Opera", y = 11 },
        new PieChartData { x = "Android", y = 12 },
    };
}

```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class PieChartData
    {
        public string xValue;
        public double yValue;
    }

```

Region

The annotation can be placed with respect to either **Series** or **Chart**.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.Annotation(an => {an
    .X("150")
    .Y("150")
    .CoordinateUnits(Syncfusion.EJ2.Charts.Units.Pixel)
    .Regions(Syncfusion.EJ2.Charts.Regions.Chart)
    .Content(ViewBag.content).Add();
}).
.LegendSettings(ls => ls.Visible(false)).Render()

```

REGION.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 37 },
        new PieChartData { x = "UC Browser", y = 17 },
        new PieChartData { x = "iPhone", y = 19 },
        new PieChartData { x = "Others", y = 4 },
        new PieChartData { x = "Opera", y = 11 },
        new PieChartData { x = "Android", y = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
}

```


Co-ordinate Units

Specifies the coordinate units of an annotation either in **Pixel** or **Point**.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
. Annotations(an => {an
    .X("iPhone")
    .Y("19")
    .CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point)
    .Regions(Syncfusion.EJ2.Charts.Regions.Series)
    .Content(ViewBag.content).Add();
}).
.LegendSettings(ls => ls.Visible(false)).Render()
```

CO-ORDINATE.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 37 },
        new PieChartData { x = "UC Browser", y = 17 },
        new PieChartData { x = "iPhone", y = 19 },
        new PieChartData { x = "Others", y = 4 },
        new PieChartData { x = "Opera", y = 11 },
        new PieChartData { x = "Android", y = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Alignment

The annotations can be moved vertically and horizontally from its default position by using **VerticalAlignment** or **HorizontalAlignment** properties. The **verticalAlignment** property takes value as **Top**, **Bottom** or **Middle** and the **horizontalAlignment** property takes value as **Near**, **Far** or **Center**.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
. Annotations(an => {an
    .X("iPhone")
    .Y("19")
    .CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point)
    .Regions(Syncfusion.EJ2.Charts.Regions.Series)
    .VerticalAlignment(Syncfusion.EJ2.Charts.Position.Top)
    .HorizontalAlignment(Syncfusion.EJ2.Charts.Position.Near)
    .Content(ViewBag.content).Add();
}).
.LegendSettings(ls => ls.Visible(false)).Render()
```

ALIGNMENT.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 37 },
        new PieChartData { x = "UC Browser", y = 17 },
        new PieChartData { x = "iPhone", y = 19 },
        new PieChartData { x = "Others", y = 4 },
        new PieChartData { x = "Opera", y = 11 },
        new PieChartData { x = "Android", y = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Tooltip in ASP.NET MVC Accumulation chart component

Tooltip for the accumulation chart can be enabled by using the [Enable](#) property.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
```

```

        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(ls => ls.Visible(false)).Render()

```

DEFAULT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Header

We can specify header for the tooltip using [Header](#) property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.Tooltip(tp => tp.Enable(true).Header("Pie Chart"))
.LegendSettings(ls => ls.Visible(false)).Render()

```

HEADER.CS

```

public ActionResult Index()
{

```

```

List<PieChartData> chartData = new List<PieChartData>
{
    new PieChartData { xValue = "Chrome", yValue = 37 },
    new PieChartData { xValue = "UC Browser", yValue = 17 },
    new PieChartData { xValue = "iPhone", yValue = 19 },
    new PieChartData { xValue = "Others", yValue = 4 },
    new PieChartData { xValue = "Opera", yValue = 11 },
    new PieChartData { xValue = "Android", yValue = 12 },
};
ViewBag.dataSource = chartData;
return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Format

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.Tooltip(tp => tp.Enable(true).Format("format: '${point.x} : 
<b>${point.y}</b>'"))
.LegendSettings(ls => ls.Visible(false)).Render()

```

FORMAT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class PieChartData
    {
        public string xValue;
        public double yValue;
    }

```

Tooltip format

Any HTML element can be displayed in the tooltip by using the [Template](#) property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.Tooltip(tp => tp.Enable(true).Template("<div id='templateWrap'
style='background-color:#bd18f9;border-radius: 3px; float: right;padding:
2px;line-height: 20px;text-align: center;'>" +
    "<img src='sun_annotation.png' />" +
    "<div style='color:white; font-family:Roboto; font-style:
medium; font-size:14px;float: right;padding: 2px;line-height: 20px;text-
align: center;padding-right:6px;'><span>${y}</span></div></div>" ))
.LegendSettings(ls => ls.Visible(false)).Render()

```

TOOLTIP-FORMAT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Fixed tooltip

By default, tooltip track the mouse movement, but you can set a fixed position for the tooltip by using the [Location](#) property.

CSHTML

```
@(Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Add();
}))
.Title("Mobile Browser Statistics")
.Tooltip(tp => tp.Enable(true).Location(lc => lc.X(200).Y(20)))
.LegendSettings(ls => ls.Visible(false)).Render()
```

TOOLTIP-FIXED.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Customization

The [Fill](#) and [Border](#) properties are used to customize the background color and border of the tooltip respectively. The [TextStyle](#) property in the tooltip is used to customize the font of the tooltip text. The [HighlightColor](#) property can be used to change the color of the data point when hovering.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
```

```

        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .Tooltip(tp => tp.Enable(true).Format(format: "${series.name}
    ${point.x} : ${point.y}")
    .Fill('#7bb4eb')
    })
    .LegendSettings(ls => ls.Visible(false)).Render()

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

To customize individual tooltip

Using [TooltipRender](#) event, you can customize a tooltip for particular point. event, you can customize a tooltip for particular point.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.Tooltip(tp => tp.Enable(true))
.PointRender("pointRender")
.LegendSettings(ls => ls.Visible(false)).Render()

```

```
<script>
    var pointRender = function (args) {
        if (args.point.index === 3) {
            args.text = args.point.xValue + ' ' + ':' + args.point.yValue +
            ' ' + ' ' + 'customtext';
            args.textStyle.color = '#f48042';
        }
    };
</script>
```

INDIVIDUAL.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Tooltip mapping name

By default, tooltip shows information of x and y value in points. You can show more information from datasource in tooltip by using the [TooltipMappingName](#) property of the tooltip. You can use the `#{point.tooltip}` as place holders to display the specified tooltip content.

CSHTML

```
@(Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .XName("xValue")
        .YName("yValue")
        .TooltipMappingName("text")
        .Name("RIO")
        .Radius("70%")
        .StartAngle(0)
        .EndAngle(360)
        .InnerRadius("0%")
        .DataSource(ViewBag.dataSource).Add();
    })
    .Tooltip(tp =>
tp.Enable(true).Format("#{point.tooltip}"))
```



```

        .LegendSettings(leg => leg.Visible(true))
        .Render()
    )

```

MAPPING.CS

```

public ActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
        new GroupingChartData { xValue = "China",          yValue =
26, text = "China: 26" },
        new GroupingChartData { xValue = "Russia",          yValue =
19, text = "Russia: 19" },
        new GroupingChartData { xValue = "Germany",          yValue =
17, text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan",          yValue =
12, text = "Japan: 12" },
        new GroupingChartData { xValue = "France",          yValue =
10, text = "France: 10" },
        new GroupingChartData { xValue = "South Korea",      yValue =
9, text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain",    yValue =
27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy",          yValue =
8, text = "Italy: 8" },
        new GroupingChartData { xValue = "Australia",        yValue =
8, text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands",      yValue =
8, text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary",          yValue =
8, text = "Hungary: 8" },
        new GroupingChartData { xValue = "Brazil",          yValue =
7, text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain",          yValue =
7, text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya",          yValue =
6, text = "Kenya: 6" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class GroupingChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Legend in ASP.NET MVC Accumulation Chart Component

As like a chart, the legend is also available for accumulation charts, which gives information about the points. By default, the legend will be placed on the right, if the width of the chart is high or will be placed on the bottom, if the height of the chart is high. Other customization features regarding the

legend element are same as the [chart legend](#). Here, the legend for a point can be collapsed by giving the empty string to the x value of the point.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(true)).Render()
```

DEFAULT.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Position and Alignment

By using the position property, you can position the legend at the **Left**, **Right**, **Top** or **Bottom** of the chart. You can also align the legend to **Center**, **Far** or **Near** of the chart using the alignment property.

CSHTML

```
@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
```

```

    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(true)
    .Position(Syncfusion.EJ2.Charts.LegendPosition.Top)).Render()

```

POSITION.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Legend Reverse

You can reverse the order of the legend items by using the [Reverse](#) property. By default, legend for the first series in the collection will be placed first.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x").YName("yValue").DataSource(ViewBag.dataSource)
    .Name("Gold").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x").YName("yValue1").DataSource(ViewBag.dataSource)
    .Name("Silver").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x").YName("yValue2").DataSource(ViewBag.dataSource)
    .Name("Brozen").Width(2).Add();
}).PrimaryYAxis(px =>
px.LabelStyle(ls=>ls.Color("transparent")).LineStyle(ls=>ls.Width(0)).MajorT
ickLines(mg=>mg.Width(0))
    .MajorGridLines(mg=>mg.Width(0))
    ).PrimaryXAxis(px => px.Interval(1)

.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg=>mg.W
idth(0))
    ).ChartArea(area => area.Border(br=>br.Color("transparent"))

```

```

        ).LegendSettings(lg => lg.reverse(true)).Title("Olympic Medal Counts
- RIO").Load("load").Render()

```

REVERSE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46, yValue1=37,
yValue2=38 },
        new ColumnChartData { x= "GBR", yValue= 27, yValue1=23,
yValue2=17 },
        new ColumnChartData { x= "CHN", yValue= 26, yValue1=18,
yValue2=26 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
    public double yValue1;
    public double yValue2;
}

```

Legend Shape

To change the legend icon shape, use the **LegendShape** property in the **Series**. By default, legend icon shape is **SeriesType**.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .LegendShape("Rectangle")
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(true)).Render()

```

LEGEND-SHAPE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },

```

```

        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Legend Size

The legend size can be changed by using the **Width** and **Height** properties of the **LegendSettings**.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(true)
.Height("28%")
.Width("50%")
.Legend.Border(ViewBag.border)).Render()

```

SIZE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string xValue;
}

```

```

        public double yValue;
    }

```

Legend Item Size

You can customize the size of the legend items by using the **ShapeHeight** and **ShapeWidth** properties.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(true)
.ShapeHeight(15)
.ShapeWidth(15)).Render()

```

ITEM-SIZE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Paging for Legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. You can view the each legend item by navigating between the pages using the navigation buttons.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")

```

```

        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(true))
    .Height("150")
    .Width("80")).Render()
)

```

PAGING.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Legend Text Wrap

When the legend text exceeds the container, the text can be wrapped by using **TextWrap** Property. End user can also wrap the legend text based on the **MaximumLabelWidth** property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")

    .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie).Add();
})
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(true))

    .MaximumLabelWidth(50).TextWrap(Syncfusion.EJ2.Charts.TextWrap.Wrap)).Render()
()

```

```
)
```

TEXTWRAP.CS

```
public IActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}
```

Enable Animation

You can customize the animation while clicking legend by setting enableAnimation as true or false using **EnableAnimation** property in Accumulation Chart.

CSHTML

```
@(Html.EJS().AccumulationChart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .XName("xValue")
        .YName("yValue")
        .Name("RIO")
        .Radius("70%")
        .StartAngle(0)
        .EndAngle(360)
        .InnerRadius("0%")
        .DataSource(ViewBag.dataSource).Add();
    })
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(leg => leg.Visible(true))
    .EnableAnimation(true)
    .Title("RIO Olympics Gold").Render()
)
```

ACCUMULATION-ANIMATION.CS

```
public IActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
```



```

        new GroupingChartData { xValue = "China",          yValue =
26, text = "China: 26" },
        new GroupingChartData { xValue = "Russia",        yValue =
19, text = "Russia: 19" },
        new GroupingChartData { xValue = "Germany",       yValue =
17, text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan",         yValue =
12, text = "Japan: 12" },
        new GroupingChartData { xValue = "France",        yValue =
10, text = "France: 10" },
        new GroupingChartData { xValue = "South Korea",   yValue =
9, text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain", yValue =
27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy",         yValue =
8, text = "Italy: 8" },
        new GroupingChartData { xValue = "Australia",     yValue =
8, text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands",   yValue =
8, text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary",       yValue =
8, text = "Hungary: 8" },
        new GroupingChartData { xValue = "Brazil",        yValue =
7, text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain",         yValue =
7, text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya",         yValue =
6, text = "Kenya: 6" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class GroupingChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Legend Title

You can set title for legend using **Title** property in **LegendSettings**. You can also customize the **FontStyle**, **Size**, **FontWeight**, **Color**, **TextAlignment**, **FontFamily**, **Opacity** and **TextOverflow** of legend title. **TitlePosition** is used to set the legend position in **Top**, **Left** and **Right** position. **MaximumTitleWidth** is used to set the width of the legend title. By default, it will be **100px**.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Add();
}

```

```

    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(true))

.Position(Syncfusion.EJ2.Charts.LegendPosition.Bottom).Title("Browsers"))
.Render()

```

TITLE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Arrow Page Navigation

By default, the page number will be enabled while legend paging. Now, you can disable that page number and also you can get left and right arrows for page navigation. You have to set `false` value to `EnablePages` to get this support.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(true))

.Position(Syncfusion.EJ2.Charts.LegendPosition.Bottom).EnablePages(false))
.Render()

```

ARROW-PAGE.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Legend Item Padding

The [ItemPadding](#) property can be used to adjust the space between the legend items.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .Add();
})
.EnableSmartLabels(true)
.Title("Mobile Browser Statistics")
.LegendSettings(ls => ls.Visible(true)

.Position(Syncfusion.EJ2.Charts.LegendPosition.Bottom).EnablePages(false).Item
emPadding(30)
.Render()

```

ITEMPADDING.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class PieChartData
    {
        public string xValue;
        public double yValue;
    }

```

Center label in ASP.NET MVC Accumulation Chart Component

Center label

Using [CenterLabel](#) it is now possible to place a label at the center of a pie or doughnut chart. To configure the default text rendered on the center label for the pie and doughnut charts, use the [Text](#) property in the [CenterLabel](#).

CSHTML

```

Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("x")
        .YName("y")
        .InnerRadius("65%").Add();
})
.CenterLabel(cl => cl.Text("Mobile<br>Browsers<br>Statistics"))
.LegendSettings(ls => ls.Visible(false)).Render()

```

CENTERLABEL.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 61.3, text = "Chrome: 61.3%"},
        new PieChartData { x = "Safari", y = 24.6, text = "Safari: 24.6%"},
        new PieChartData { x = "Edge", y = 5.0, text = "Edge: 5.0%"},
        new PieChartData { x = "Samsung Internet", y = 2.7, text = "Samsung Internet: 2.7%"},
        new PieChartData { x = "Firefox", y = 2.6, text = "Firefox: 2.6%"},
        new PieChartData { x = "Others", y = 3.6, text = "Others: 3.6%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PieChartData
{
    public string x;
    public double y;
    public string text;
}

```

Hover text

The default text in the center label can be changed when the mouse pointer hovers over the pie and doughnut charts slice using the [HoverTextFormat](#) property.

CSHTML

```
Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("x")
        .YName("y")
        .InnerRadius("65%").Add();
})
.CenterLabel(cl => cl.Text("Mobile<br>Browsers<br>Statistics")
.HoverTextFormat("${point.x} <br> Browser Share <br>
${point.y}%"))
.LegendSettings(ls => ls.Visible(false)).Render()
```

HOVERTEXT.CS

```
public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 61.3, text = "Chrome:
61.3%"},
        new PieChartData { x = "Safari", y = 24.6, text = "Safari:
24.6%"},
        new PieChartData { x = "Edge", y = 5.0, text = "Edge:
5.0%"},
        new PieChartData { x = "Samsung Internet", y = 2.7, text =
"Samsung Internet: 2.7%"},
        new PieChartData { x = "Firefox", y = 2.6, text = "Firefox:
2.6%"},
        new PieChartData { x = "Others", y = 3.6, text = "Others:
3.6%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string x;
    public double y;
    public string text;
}
```

Customization

Customize the center label text using the [TextStyle](#) property.

CSHTML

```
Html.EJS().AccumulationChart("container").Series(series =>
{
```

```

        series.DataSource(ViewBag.dataSource)
            .XName("x")
            .YName("y")
            .InnerRadius("65%").Add();
    })
    .CenterLabel(cl => cl.Text("Mobile<br>Browsers<br>Statistics"))
    .HoverTextFormat("${point.x} <br> Browser Share <br>
    ${point.y}%")
    .TextStyle(ts =>
    ts.FontWeight("900").Size("15px").Color("grey").FontFamily("Roboto").FontStyle("Italic"))
    .LegendSettings(ls => ls.Visible(false)).Render()

```

CUSTOMIZATION.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { x = "Chrome", y = 61.3, text = "Chrome:
61.3%"},
        new PieChartData { x = "Safari", y = 24.6, text = "Safari:
24.6%"},
        new PieChartData { x = "Edge", y = 5.0, text = "Edge:
5.0%"},
        new PieChartData { x = "Samsung Internet", y = 2.7, text =
"Samsung Internet: 2.7%"},
        new PieChartData { x = "Firefox", y = 2.6, text = "Firefox:
2.6%"},
        new PieChartData { x = "Others", y = 3.6, text = "Others:
3.6%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string x;
    public double y;
    public string text;
}

```

<!-- markdownlint-disable MD036 -->

Title

Accumulation Chart can be given a title using [Title](#) property, to show the information about the data plotted.

CSHTML

```

@Html.EJS().AccumulationChart("container").Title("Oil
and other liquid imports in USA").LegendSettings(ls =>
ls.Visible(false)).Series(series =>
{

```

```

series.DataLabel(dl =>
dl.Visible(true).Name("text"))
                .XName("xValue")
                .YName("yValue")
                .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
                .DataSource(ViewBag.dataSource).Add();
}).Render()

```

TITLE.CS

```

public ActionResult Pie()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Saudi Arabia", yValue = 58,
text = "58%"},
        new PieChartData { xValue = "Persian Gulf", yValue = 15,
text = "15%"},
        new PieChartData { xValue = "Canada", yValue = 13, text =
"13%"},
        new PieChartData { xValue = "Venezuela", yValue = 8 , text =
"8%"},
        new PieChartData { xValue = "Mexico", yValue = 3 , text =
"3%"},
        new PieChartData { xValue = "Russia", yValue = 2, text =
"2%"},
        new PieChartData { xValue = "Miscellaneous", yValue = 1,
text = "1%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Title Customization

Accumulation Chart can be customized a title using [TitleStyle](#) property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Title("Oil
and other liquid imports in USA").SubTitle("In the year 2014 - 2015")
                .LegendSettings(ls =>
ls.Visible(false)).TitleStyle(Sts =>
Sts.FontFamily("Arial").FontStyle("italic").FontWeight("regular").Color("#E2
7F2D").Size("23px")).
                Series(series =>
{

```

```

series.DataLabel (dl =>
dl.Visible(true).Name("text"))
                .XName("xValue")
                .YName("yValue")
                .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
                .DataSource(ViewBag.dataSource).Add();
    }).Render()

```

TITLE.CS

```

public ActionResult Pie()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Saudi Arabia", yValue = 58,
text = "58%"},
        new PieChartData { xValue = "Persian Gulf", yValue = 15,
text = "15%"},
        new PieChartData { xValue = "Canada", yValue = 13, text =
"13%"},
        new PieChartData { xValue = "Venezuela", yValue = 8 , text =
"8%"},
        new PieChartData { xValue = "Mexico", yValue = 3 , text =
"3%"},
        new PieChartData { xValue = "Russia", yValue = 2, text =
"2%"},
        new PieChartData { xValue = "Miscellaneous", yValue = 1,
text = "1%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

SubTitle

Accumulation Chart can be given a subtitle using [SubTitle](#) property, to show the information about the data plotted.

CSHTML

```

@Html.EJS().AccumulationChart("container").Title("Oil
and other liquid imports in USA").SubTitle("In the year 2014 -
2015").LegendSettings(ls => ls.Visible(false)).Series(series =>
{
    series.DataLabel (dl =>
dl.Visible(true).Name("text"))
                .XName("xValue")

```



```

.YName("yValue")
.Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
.DataSource(ViewBag.dataSource).Add();
}).Render()

```

TITLE.CS

```

public ActionResult Pie()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Saudi Arabia", yValue = 58,
text = "58%"},
        new PieChartData { xValue = "Persian Gulf", yValue = 15,
text = "15%"},
        new PieChartData { xValue = "Canada", yValue = 13, text =
"13%"},
        new PieChartData { xValue = "Venezuela", yValue = 8 , text =
"8%"},
        new PieChartData { xValue = "Mexico", yValue = 3 , text =
"3%"},
        new PieChartData { xValue = "Russia", yValue = 2, text =
"2%"},
        new PieChartData { xValue = "Miscellaneous", yValue = 1,
text = "1%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

SubTitle Customization

Accumulation Chart can be customized a subtitle using [SubTitleStyle](#) property.

CSHTML

```

@Html.EJS().AccumulationChart("container").Title("Oil
and other liquid imports in USA").SubTitle("In the year 2014 - 2015")
    .LegendSettings(ls =>
ls.Visible(false)).SubTitleStyle(Sts =>
Sts.FontFamily("Arial").FontStyle("italic").FontWeight("regular").Color("#E2
7F2D").Size("13px")).
    Series(series =>
    {
        series.DataLabel(dl =>
dl.Visible(true).Name("text"))
        .XName("xValue")
        .YName("yValue")

```

```

        .Name("Browser")

.Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .DataSource(ViewBag.dataSource).Add();
    }).Render()

```

TITLE.CS

```

public ActionResult Pie()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Saudi Arabia", yValue = 58,
text = "58%"},
        new PieChartData { xValue = "Persian Gulf", yValue = 15,
text = "15%"},
        new PieChartData { xValue = "Canada", yValue = 13, text =
"13%"},
        new PieChartData { xValue = "Venezuela", yValue = 8 , text =
"8%"},
        new PieChartData { xValue = "Mexico", yValue = 3 , text =
"3%"},
        new PieChartData { xValue = "Russia", yValue = 2, text =
"2%"},
        new PieChartData { xValue = "Miscellaneous", yValue = 1,
text = "1%"}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Print and Export**Print**

The rendered chart can be printed directly from the browser by calling the public method print.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")
        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
})
.EnableSmartLabels(true)
.ChartMouseClicked("chartMouseClicked")
.Title("Mobile Browser Statistics")

```

```

        .LegendSettings(ls => ls.Visible(false)).Render()
<div class="col-lg-3 property-section">
    <table id="property" title="Properties" style="width: 100%">
        <tr id="button-control" style="height: 50px">
            <td align='center'>
                <div>
                    @Html.EJS().Button("togglebtn").IconCss("e-icons
e-play-icon").Content("Print").CssClass("e-flat").Render()
                </div>
            </td>
        </tr>
    </table>
</div>
</div>
<script>
    var chartMouseClicked = function (args) {
        if (args.target.indexOf('print') > -1) {
            chart.print();
        }
    };
</script>

```

PRINT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

Export

The rendered chart can be exported to **Image**(jpeg or png) or **SVG** or **PDF** format by using the export method. Input parameters for this method are **Export Type** for **format** and **fileName** of result.

CSHTML

```

@Html.EJS().AccumulationChart("container").Series(series =>
{
    series.DataSource(ViewBag.dataSource)
        .XName("xValue")

```

```

        .YName("yValue")
        .Name("Browser")
        .Type(Syncfusion.EJ2.Charts.AccumulationType.Pie)
        .ExplodeIndex(0).Add();
    })
    .EnableSmartLabels(true)
    .Title("Mobile Browser Statistics")
    .LegendSettings(ls => ls.Visible(false)).Render()
<div class="col-lg-3 property-section">
    <table id="property" title="Properties" style="width: 100%">
        <tr id="button-control" style="height: 50px">
            <td align="center">
                <div>
                    @Html.EJS().Button("togglebtn").IconCss("e-icons
e-play-icon").Content("Export").CssClass("e-flat").Render()
                </div>
            </td>
        </tr>
    </table>
</div>
</div>
<script>
    document.getElementById('togglebtn').onclick = function () {
        var chart = document.getElementById('export-
container').ej2_instances[0];
        var fileName = (document.getElementById('fileName')).value;
        chart.export(mode, fileName);
    };
</script>

```

EXPORT.CS

```

public ActionResult Index()
{
    List<PieChartData> chartData = new List<PieChartData>
    {
        new PieChartData { xValue = "Chrome", yValue = 37 },
        new PieChartData { xValue = "UC Browser", yValue = 17 },
        new PieChartData { xValue = "iPhone", yValue = 19 },
        new PieChartData { xValue = "Others", yValue = 4 },
        new PieChartData { xValue = "Opera", yValue = 11 },
        new PieChartData { xValue = "Android", yValue = 12 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PieChartData
{
    public string xValue;
    public double yValue;
}

```

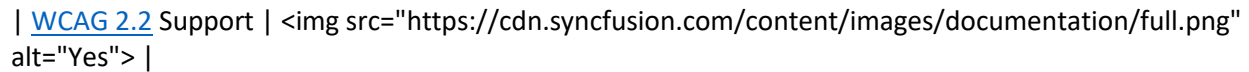
Accessibility in ASP.NET MVC Accumulation chart component

The Accumulation chart component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Accumulation chart component is outlined below.

| Accessibility Criteria | Compatibility |

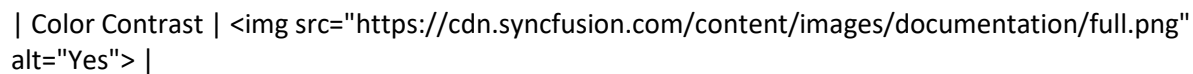
| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Accumulation chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Accumulation chart component:

- `img (role)`

- button (role)
- region (role)
- aria-label (attribute)
- aria-hidden (attribute)
- aria-pressed (attribute)

Keyboard interaction

The Accumulation chart component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Accumulation chart component.

- | Press | To do this |
- | --- | --- |
- | Alt + J | Moves the focus to the Accumulation chart element. |
- | Tab | Moves the focus to the next element in the Accumulation chart. |
- | Shift + Tab | Moves the focus to the previous element in the Accumulation chart. |
- | Down Arrow | Moves the focus to the data point left side from the selected point. |
- | Up Arrow | Moves the focus to the data point right side from the selected point. |
- | Down/Left Arrow | Moves the focus to the legend left side from the selected legend. |
- | Up/Right Arrow | Moves the focus to the legend right side from the selected legend. |
- | Enter/Space | Toggles the visibility of the corresponding series. |
- | Ctrl + P | Prints the Accumulation chart. |

Ensuring accessibility

The Accumulation chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Accumulation chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Accumulation chart component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Migration from Essential JS 1

This article describes the API migration process of Accumulation chart component from Essential JS 1 to Essential JS 2.

Accumulation Chart

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Annotations	Property:Annotations@ (Html.EJ().Chart("chartContainer").Annotations())	Property:annotations@ (Html.EJS().AccumulationChart("container").Annotations())
Background	Property:Background@ (Html.EJ().Chart("chartContainer").Background())	Property:Background@ (Html.EJS().AccumulationChart("container").Background())
Border	Property:Border@ (Html.EJ().Chart("chartContainer").Border(b => b.Color("red").Width("2")))	Property:Border@ (Html.EJS().AccumulationChart("container").Background()).Border(b => b.Color("red").Width("2"))
DataSource	Not applicable	Property:DataSource@ (Html.EJS().AccumulationChart("container").DataSource())
Animation after legend click	Not applicable	Property:EnableAnimation@ (Html.EJS().AccumulationChart("container").EnableAnimation(true))
Persisting component's state between page reloads.	Not applicable	Property:EnablePersistence@ (Html.EJS().AccumulationChart("container").EnablePersistence(true))
Enabling smart labels	Property:Series.EnableSmartLabels@ (Html.EJ().Chart("chartContainer").Series(s => s.EnableSmartLabels(true)))	Property:EnableSmartLabels@ (Html.EJS().AccumulationChart("container").Series(s => s.EnableSmartLabels(true)))
Height of Chart	Property:Size.Height@ (Html.EJ().Chart("chartContainer").Size(s => s.Height("400")))	Property:Height@ (Html.EJS().AccumulationChart("container").Height("400"))
Multi selection	Property:SelectionSettings.Type@ (Html.EJ().Chart("chartContainer").SelectionSettings(s => s.Type("multiple")))	Property:IsMultiSelect@ (Html.EJS().AccumulationChart("container").IsMultiSelect(true))
Legend Settings	Property:Legend@ (Html.EJ().Chart("chartContainer").Legend())	Property:LegendSettings@ (Html.EJS().AccumulationChart("container").LegendSettings(true))
Margin for the chart	Property:Margin@ (Html.EJ().Chart("chartContainer").Margin(m => m.Top("10").Bottom("10")))	Property:Margin@ (Html.EJS().AccumulationChart("container").Margin(m => m.Top("10").Bottom("10")))
Selected DataIndexes	Property:SelectedDataPointIndexes@ (Html.EJ().Chart("chartContainer").SelectedDataPointIndexes())	Property:SelectedDataIndexes@ (Html.EJS().AccumulationChart("container").SelectedDataIndexes(s => s.Series("0").Point("1")))

Selection Mode	Property:SelectionSettings.Mode@(Html.EJ().Chart("chartContainer").SelectionSettings(s => s.Mode("point")))	Property:selectionMode@(Html.EJS().AccumulationChart("container").SelectionSettings(s => s.Mode("Point")))
Series	Property:Series@(Html.EJ().Chart("chartContainer").Series())	Property:Series@(Html.EJS().AccumulationChart("container").Series())
Title text	Property:Title.Text@(Html.EJ().Chart("chartContainer").Title(t => t.Text("pie chart")))	Property:Title@(Html.EJS().AccumulationChart("container").Title(t => t.Text("pie chart")))
Title Styles	Property:TitleStyle@(Html.EJ().Chart("chartContainer").TitleStyle(t => t.FontFamily("SegoeUI")))	Property:TitleStyle@(Html.EJS().AccumulationChart("container").TitleStyle(t => t.FontFamily("SegoeUI")))
Sub Title text	Property:subTitle.text@(Html.EJ().Chart("Container").subTitle(t => t.Text("chart")))	Property:subTitle@(Html.EJS().AccumulationChart("Container").subTitle(t => t.Text("chart")))
Sub title Styles	Property:Title@(Html.EJ().Chart("chartContainer").SubTitleStyle(s => s.Font()))	Property:SubTitleStyle@(Html.EJS().AccumulationChart("container").SubTitleStyle(s => s.Font()))
tooltip	Property:Series.ToolTip@(Html.EJ().Chart("chartContainer").Series(s => s.ToolTip()))	Property:Tooltip@(Html.EJS().AccumulationChart("container").Series(s => s.ToolTip()))
Width of Chart	Property:Size.Width@(Html.EJ().Chart("chartContainer").Size(s => s.Width("400")))	Property:Width@(Html.EJS().AccumulationChart("container").Size(s => s.Width("400")))

Annotation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Content	Property:Annotations.Content@(Html.EJ().Chart("chartContainer").Annotations(a => a.Content("divID")))	Property:Annotations.Content@(Html.EJS().AccumulationChart("container").Annotations(a => a.Content("divID")))
coordinateUnits	Property:Annotations.CoordinateUnit@(Html.EJ().Chart("chartContainer").Annotations(a => a.CoordinateUnit("pixel")))	Property:Annotations.CoordinateUnit@(Html.EJS().AccumulationChart("container").Annotations(a => a.CoordinateUnit("pixel")))
description	Not Applicable	Property:Description@(Html.EJS().AccumulationChart("container").Annotations(a => a.Description("")))
horizontalAlignment for	Property:Annotations.HorizontalAlignment@(Html.EJ().Chart("chartContainer").Annotations(a => a.HorizontalAlignment("middle")))	Property:annotations.horizontalAlignment@(Html.EJS().AccumulationChart("container").Annotations(a => a.HorizontalAlignment("center")))

annotation		
margin for annotation	Property:Annotations.Margin @(Html.EJ().Chart("chartContainer").Annotations(a => a.Margin(m => m.Top("10").Bottom("10"))))	Not applicable
Opacity for annotation	Property:Annotations.Opacity @(Html.EJ().Chart("chartContainer").Annotations(a => a.Opacity("0.5")))	Not applicable
Region for annotation with respect to chart or series	Property:Annotations.Region @(Html.EJ().Chart("chartContainer").Annotations(a => a.Region("chart")))	Property:Annotations.Region@(Html.EJS().AccumulationChart("container").Annotations(a => a.Region("chart")))
verticalAlignment for annotation	Property:Annotations.VerticalAlignment @(Html.EJ().Chart("chartContainer").Annotations(a => a.VerticalAlignment("middle")))	Property:Annotations.VerticalAlignment@(Html.EJS().AccumulationChart("container").Annotations(a => a.VerticalAlignment("center")))
Visibility of annotations	Property:Annotations.Visible @(Html.EJ().Chart("chartContainer").Annotations(a => a.Visible(true)))	Not applicable
X offset for annotation	Property:Annotations.X @(Html.EJ().Chart("chartContainer").Annotations(a => a.X("50")))	Property:Annotations.X @(Html.EJS().AccumulationChart("container").Annotations(a => a.X("50")))
Y offset for annotation	Property:Annotations.Y @(Html.EJ().Chart("chartContainer").Annotations(a => a.Y("150")))	Property:Annotations.Y @(Html.EJS().AccumulationChart("container").Annotations(a => a.Y("150")))

Series

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
series	Property:Series @(Html.EJ().Chart("chartContainer").Series())	Property:Series @(Html.EJS().AccumulationChart("container").Series())

animation for series	<code>Property:EnableAnimation@(Html.EJ().Chart("chartContainer").Series(s => s.EnableAnimation(true)))</code>	<code>Property:Animation.Enable@(Html.EJS().AccumulationChart("container").Series(s => s.Animation(a => a.Enable(true))))</code>
animation duration for series	<code>Property:AnimationDuration@(Html.EJ().Chart("chartContainer").Series(s => s.AnimationDuration("1000")))</code>	<code>Property:Animation.Duration@(Html.EJS().AccumulationChart("container").Series(s => s.Animation(a => a.Duration("1000"))))</code>
animation delay for series	Not Applicable	<code>Property:Animation.Delay@(Html.EJS().AccumulationChart("container").Series(s => s.Animation(a => a.Delay("1000"))))</code>
Border for series	<code>Property:Border@(Html.EJ().Chart("chartContainer").Series(s => s.Border(b => b.Color("red").Width("2"))))</code>	<code>Property:Border@(Html.EJS().AccumulationChart("container").Series(s => s.Border(b => b.Color("red").Width("2"))))</code>
DataLabel for series	<code>Property:DataLabel@(Html.EJ().Chart("chartContainer").Series(s => s.DataLabel()))</code>	<code>Property:DataLabel@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel()))</code>
DataSource for series	<code>Property:DataSource@(Html.EJ().Chart("chartContainer").Series(s => s.DataSource()))</code>	<code>Property:DataSource@(Html.EJS().AccumulationChart("container").Series(s => s.DataSource()))</code>
enable Tooltip for series	<code>Property:Tooltip.Visible@(Html.EJ().Chart("chartContainer").Series(s => s.Tooltip(t => t.Visible(true))))</code>	<code>Property:EnableTooltip@(Html.EJS().AccumulationChart("container").Series(s => s.EnableTooltip(true)))</code>
start angle	<code>Property:StartAngle@(Html.EJ().Chart("chartContainer").Series(s => s.StartAngle("90")))</code>	<code>Property:StartAngle@(Html.EJS().AccumulationChart("container").Series(s => s.StartAngle("90")))</code>
end angle	<code>Property:EndAngle@(Html.EJ().Chart("chartContainer").Series(s => s.EndAngle("190")))</code>	<code>Property:EndAngle@(Html.EJS().AccumulationChart("container").Series(s => s.EndAngle("190")))</code>
explode	<code>Property:Explode@(Html.EJS().AccumulationChart("container").Series(s => s.Explode(true)))</code>	<code>Property:Explode@(Html.EJS().AccumulationChart("container").Series(s => s.Explode(true)))</code>
explodeAll	<code>Property:ExplodeAll@(Html.EJ().Chart("chartContainer").Series(s => s.ExplodeAll(true)))</code>	<code>Property:ExplodeAll@(Html.EJS().AccumulationChart("container").Series(s => s.ExplodeAll(true)))</code>

explodeIndex	Property:ExplodeIndex@(Html.EJ().Chart("chartContainer").Series(s => s.ExplodeIndex("0")))	Property:ExplodeIndex@(Html.EJS().AccumulationChart("container").Series(s => s.EExplodeIndex("1")))
explodeOffset	Property:ExplodeOffset@(Html.EJ().Chart("chartContainer").Series(s => s.ExplodeOffset("30%")))	Property:ExplodeOffset@(Html.EJS().AccumulationChart("container").Series(s => s.ExplodeOffset("30%")))
gapRatio	Property:GapRatio@(Html.EJ().Chart("chartContainer").Series(s => s.GapRatio("0.6")))	Property:GapRatio@(Html.EJS().AccumulationChart("container").Series(s => s.GapRatio("0.6")))
gapWidth	Property:GapWidth@(Html.EJ().Chart("chartContainer").Series(s => s.GapWidth("5")))	Not Applicable
inner radius of the accumulation chart	Property:InnerRadius @(Html.EJ().Chart("chartContainer").Series(s => s.InnerRadius("30%")))	Property:innerRadius @(Html.EJS().AccumulationChart("container").Series(s => s.InnerRadius("30%")))
Legend shape of the series	Not applicable	Property:LegendShape@(Html.EJS().AccumulationChart("container").Series(s => s.LegendShape("Rectangle")))
name of the series	Property:Name@(Html.EJ().Chart("chartContainer").Series(s => s.Name("")))	Property:Name@(Html.EJS().AccumulationChart("container").Series(s => s.Name("")))
neck height for funnel series	Property:NeckHeight@(Html.EJ().Chart("chartContainer").Series(s => s.NeckHeight("50")))	Property:NeckHeight@(Html.EJS().AccumulationChart("container").Series(s => s.NeckHeight("50")))
neck width for funnel series	Property:NeckWidth@(Html.EJ().Chart("chartContainer").Series(s => s.NeckWidth("50")))	Property:NeckWidth@(Html.EJS().AccumulationChart("container").Series(s => s.NeckWidth("50")))
opacity for series	Property:Opacity@(Html.EJ().Chart("chartContainer").Series(s => s.Opacity("0.5")))	Property:Opacity@(Html.EJS().AccumulationChart("container").Series(s => s.Opacity("0.5")))
palettes for series	Property:Palette@(Html.EJ().Chart("chartContainer").Series(s => s.Palette()))	Property:Palettes@(Html.EJS().AccumulationChart("container").Series(s => s.Palette()))

point color mapping name for series	<code>Property:PointColorMappingName (Html.EJ().Chart("chartContainer").Series(s => s.PointColorMappingName("color")))</code>	<code>Property:PointColorMapping@ (Html.EJS().AccumulationChart("container").Series(s => s.PointColorMapping("color")))</code>
Mode of pyramid series	<code>Property:PyramidMode@ (Html.EJ().Chart("chartContainer").Series(s => s.PyramidMode("Surface")))</code>	<code>Property:PyramidMode@ (Html.EJS().AccumulationChart("container").Series(s => s.PyramidMode("Linear")))</code>
query for data source for series	<code>Property:Query@ (Html.EJ().Chart("chartContainer").Series(s => s.Query()))</code>	<code>Property:Query@ (Html.EJS().AccumulationChart("container").Series(s => s.Query()))</code>
Radius of Pie series	<code>Property:PieCoefficient@ (Html.EJ().Chart("chartContainer").Series(s => s.PieCoefficient("0.5")))</code>	<code>Property:Radius@ (Html.EJS().AccumulationChart("container").Series(s => s.Radius("0.5")))</code>
Selection Style of Accumulation chart	Not applicable	<code>Property:SelectionMode@ (Html.EJS().AccumulationChart("container").Series(s => s.SelectionStyle()))</code>
tooltip Mapping name	Not applicable	<code>Property:TooltipMappingName (Html.EJS().AccumulationChart("container").Series(s => s.TooltipMappingName()))</code>
Type of series	<code>Property:Type@ (Html.EJ().Chart("chartContainer").Series(s => s.Type("")))</code>	<code>Property:Type@ (Html.EJS().AccumulationChart("container").Series(s => s.Type("")))</code>
Name of the property in the data source that	<code>Property:XName@ (Html.EJ().Chart("chartContainer").Series(s => s.XName("x")))</code>	<code>Property:XName@ (Html.EJS().AccumulationChart("container").Series(s => s.XName("x")))</code>

contains x value for the series.		
Name of the property in the datasource that contains x value for the series.	Property:YName@(Html.EJ().Chart("chartContainer").Series(s => s.YName("y")))	Property:YName@(Html.EJS().AccumulationChart("container").Series(s => s.YName("y")))
Width of funnel series	Property:FunnelWidth@(Html.EJ().Chart("chartContainer").Series(s => s.FunnelWidth("100")))	Property:Width@(Html.EJS().AccumulationChart("container").Series(s => s.Width("100")))
Grouping	Not Applicable	Property:GroupTo@(Html.EJS().AccumulationChart("container").Series(s => s.GroupTo("1")))
Group Mode	Not Applicable	Property:GroupMode@(Html.EJS().AccumulationChart("container").Series(s => s.GroupMode("Point")))

DataLabel

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
dataLabel	Property:Series.Marker.DataLabel@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.Visible(true)))))	Property:Series.DataLabel@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Visible(true)))))
border of dataLabel	Property:Series.Marker.DataLabel.Border@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.Border(b => b.Color("red").Width("2")))))	Property:Border@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Border(b => b.Color("red").Width("2")))))

connector style for data Label connector line	<code>Property:ConnectorLine@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.ConnectorLine(c => c.Type("Curve").Width("2"))))))</code>	<code>Property:connectorStyle@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.ConnectorLine(c => c.Type("Curve").Width("2"))))))</code>
Fill for data Label	<code>Property:Fill@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.Fill("red")))))</code>	<code>Property:Fill@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Fill("red")))))</code>
font for data Label	<code>Property:Font@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.Font()))))</code>	<code>Property:font@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Font())))</code>
position	<code>Property:Position@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.Position("Inside")))))</code>	<code>Property:Position@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Position("Outside")))))</code>
Rounded corner radius X	Not Applicable	<code>Property:DataLabel.Rx@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Rx("10"))))</code>
Rounded corner radius Y	Not Applicable	<code>Property:DataLabel.Ry@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Ry("10"))))</code>
HTML template	<code>Property:DataLabel.Template@(Html.EJ().Chart("chartContainer").Series(s => s.Marker(m => m.DataLabel(d => d.Template("chart"))))</code>	<code>Property:DataLabel.Template@(Html.EJS().AccumulationChart("container").Series(s => s.DataLabel(d => d.Template("chart"))))</code>

e in data Label		
-----------------------	--	--

Legend

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Default legend	<code>Property:Visible@(Html.EJ().Chart("chartContainer").Legend(l => l.Visible(true)))</code>	<code>Property:Visible@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Visible(true)))</code>
Legend height	<code>Property:Size.Height@(Html.EJ().Chart("chartContainer").Legend(l => l.Size(s => s.Height("10"))))</code>	<code>Property:Height@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Height("10")))</code>
Legend width	<code>Property:Size.Width@(Html.EJ().Chart("chartContainer").Legend(l => l.Size(s => s.Width("10"))))</code>	<code>Property:Width@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Width("10")))</code>
Legend location in chart	<code>Property:Location@(Html.EJ().Chart("chartContainer").Legend(l => l.Location(loc => loc.X("10").Y("20"))))</code>	<code>Property:Location@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Location(loc => loc.X("10").Y("20"))))</code>
Legend position in chart	<code>Property:Position@(Html.EJ().Chart("chartContainer").Legend(l => l.Position("Top")))</code>	<code>Property:Position@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Position("Top")))</code>
Legend padding	Not applicable	<code>Property:Padding@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Padding("8")))</code>
Legend alignment	<code>Property:Alignment@(Html.EJ().Chart("chartContainer").Legend(l => l.Alignment("center")))</code>	<code>Property:Alignment@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Alignment("center")))</code>

text style for legend	<code>Property:Font@(Html.EJ().Chart("chartContainer").Legend(l => l.Font()))</code>	<code>Property:TextStyle@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Font()))</code>
shape height of legend	<code>Property:ItemStyle.Height@(Html.EJ().Chart("chartContainer").Legend(l => l.ItemStyle(i => i.Height("10"))))</code>	<code>Property:ShapeHeight@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.ShapeHeight("10")))</code>
shape width of legend	<code>Property:ItemStyle.Width@(Html.EJ().Chart("chartContainer").Legend(l => l.ItemStyle(i => i.Width("10"))))</code>	<code>Property:ShapeWidth@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.ShapeWidth("10")))</code>
shape border of legend	<code>Property:ItemStyle.Border@(Html.EJ().Chart("chartContainer").Legend(l => l.ItemStyle(i => i.Border()))</code>	Not Applicable
shape padding of legend	<code>Property:ItemPadding@(Html.EJ().Chart("chartContainer").Legend(l => l.ItemPadding("10")))</code>	<code>Property:ShapePadding@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.ShapePadding("10")))</code>
Background of legend	<code>Property:Background@(Html.EJ().Chart("chartContainer").Legend(l => l.Background("transparent")))</code>	<code>Property:Background@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Background("transparent")))</code>
Opacity of legend	<code>Property:Opacity@(Html.EJ().Chart("chartContainer").Legend(l => l.Opacity("0.5")))</code>	<code>Property:Opacity@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Opacity("0.5")))</code>
Toggle visibility of series	<code>Property:ToggleSeriesVisibility@(Html.EJ().Chart("chartContainer").Legend(l => l.ToggleSeriesVisibility("true")))</code>	<code>Property:ToggleVisibility@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.ToggleVisibility("true")))</code>

s while legend click		
Title for legend	<code>Property:Title@(Html.EJ().Chart("chartContainer").Legend(l => l.Title(t => t.Text("chartText"))))</code>	Not applicable
Text Overflow for legend	<code>Property:TextOverflow@(Html.EJ().Chart("chartContainer").Legend(l => l.TextOverflow("Trim")))</code>	<code>Property:TextStyle.TextOverflow@(Html.EJS().AccumulationChart("container").LegendSettings(l => l.Text(t => t.TextOverflow("trim"))))</code>
Text width for legend while setting text overflow	<code>Property:TextWidth@(Html.EJ().Chart("chartContainer").Legend(l => l.TextWidth("20")))</code>	Not applicable
Scroll bar for legend	<code>Property:EnableScrollBar@(Html.EJ().Chart("chartContainer").Legend(l => l.EnableScrollBar(true)))</code>	Not applicable
Row count for legend	<code>Property:RowCount@(Html.EJ().Chart("chartContainer").Legend(l => l.RowCount("3")))</code>	Not applicable
Column count for legend	<code>Property:ColumnCount@(Html.EJ().Chart("chartContainer").Legend(l => l.ColumnCount("2")))</code>	Not applicable

Color for legend items	Property:Fill@ (Html.EJ().Chart("chartContainer").Legend(1 => 1.Fill("red")))	Not applicable
------------------------	---	----------------

Methods

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
animation for series	Property:Chart.Animate@ (Html.EJ().Chart("chartContainer").Animate())	Not applicable
Redraw for chart	Property:Chart.Redraw@ (Html.EJ().Chart("chartContainer").Redraw())	Property:Chart.Refresh()@ (Html.EJS().AccumulationChart("container").Refresh())
Export	Property:Chart.Export()@ (Html.EJ().Chart("chartContainer").Export())	Property:Chart.Export()@ (Html.EJS().AccumulationChart("container").Export())
Print	Property:Chart.Print()@ (Html.EJ().Chart("chartContainer").Print())	Property:Chart.Print()@ (Html.EJS().AccumulationChart("container").Print())
AddSeries	Not Applicable	Property:Chart.AddSeries()@ (Html.EJS().AccumulationChart("container").AddSeries())
RemoveSeries	Not Applicable	Property:Chart.RemoveSeries()@ (Html.EJS().AccumulationChart("container").RemoveSeries())

Events

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Fire on annotation click	Property:AnnotationClick@ (Html.EJ().Chart("chartContainer").AnnotationClick())	Not applicable

Fire s after r ani mati on	Property:AnimationComplete@ (Html.EJ() .Chart ("chartContainer") .AnnotationComplete ())	Property:AnimationComplete()@ (Html.EJS() .AccumulationChart ("container") .AnnotationComplete ())
Fire s on after r char t resiz e	Property:AfterResize@ (Html.EJ() .Chart ("chartContainer") .AfterResize ())	Not applicable
Fire s on befo re char t resiz e	Property:BeforeResize@ (Html.EJ() .Chart ("chartContainer") .BeforeResize ())	Property:Resized@ (Html.EJS() .AccumulationChart ("container") .Resized ())
Fire s on char t click	Property:ChartClick@ (Html.EJ() .Chart ("chartContainer") .ChartClick ())	Property:ChartMouseClicked@ (Html.EJS() .AccumulationChart ("container") .ChartMouseClicked ())
Fire s on char t mou se mov e	Property:ChartMouseMove@ (Html.EJ() .Chart ("chartContainer") .ChartMouseMove ())	Property:ChartMouseMove@ (Html.EJS() .AccumulationChart ("container") .ChartMouseClicked ())
Fire s on char t mou se leav e	Property:ChartMouseLeave\$ ("#chart") .ejChart ({@ (Html.EJ() .Chart ("chartContainer") .ChartMouseLeave ())	Property:chartMouseLeave@ (Html.EJS() .AccumulationChart ("container") .ChartMouseLeave ())

Fire s on befo re char t dou ble click	Property:ChartDoubleClick@(Html.EJ().Chart("chartContainer").ChartDoubleClick())	Not applicable
Fire s on char t mou se up	Not Applicable	Property:ChartmouseUp@(Html.EJS().AccumulationChart("container").ChartmouseUp())
Fire s on char t mou se dow n	Not Applicable	Property:ChartmouseDown@(Html.EJS().AccumulationChart("container").ChartmouseDown())
Fire s duri ng the calc ulati on of char t area boun ds. You can use this eve nt to cust	Property:ChartAreaBoundsCalculate@(Html.EJ().Chart("chartContainer").ChartAreaBoundsCalculate())	Not applicable

omize the bounds of chart area		
Fire when chart is destroyed completely.	<code>Property:Destroy@(Html.EJ().Chart("chartContainer").Destroy())</code>	Not applicable
Fire after chart is created.	<code>Property:Create@(Html.EJ().Chart("chartContainer").Create())</code>	<code>Property:Loaded@(Html.EJS().AccumulationChart("container").Loaded())</code>
Fire before rendering the data labels.	<code>Property:DisplayTextRendering@(Html.EJ().Chart("chartContainer").DisplayTextRendering())</code>	<code>Property:TextRender@(Html.EJS().AccumulationChart("container").TextRender())</code>
Fire on clicking the	<code>Property:LegendItemClick@(Html.EJ().Chart("chartContainer").LegendItemClick())</code>	Not applicable

legend item.		
Fire when moving mouse over legend item.	Property:LegendItemMouseMove@ (Html.EJ() .Chart ("chartContainer") .LegendItemMouseMove ())	Not applicable
Fire before rendering the legend item.	Property:LegendItemRendering@ (Html.EJ() .Chart ("chartContainer") .LegendItemRendering ())	Property:LegendRender@ (Html.EJS() .AccumulationChart ("container") .LegendRender ())
Fire before loading the chart.	Property:Load@ (Html.EJ() .Chart ("chartContainer") .Load ())	Property:Load@ (Html.EJS() .AccumulationChart ("container") .Load ())
Fire on clicking a point in	Property:PointRegionClick@ (Html.EJ() .Chart ("chartContainer") .PointRegionClick ())	Property:PointClick@ (Html.EJS() .AccumulationChart ("container") .PointClick ())

chart.		
Fire s when mouse is moved over a point.	Property:PointRegionMouseMove @(Html.EJ().Chart("chartContainer").PointRegionMouseMove())	Property:PointMove @(Html.EJS().AccumulationChart("container").PointMove())
Fire s before rendering chart.	Property:PreRender @(Html.EJ().Chart("chartContainer").PreRender())	Not applicable
Fire s when point render.	Not Applicable	Property:PointRender @(Html.EJS().AccumulationChart("container").PointRender())
Fire s before rendering a series.	Property:SeriesRendering @(Html.EJ().Chart("chartContainer").SeriesRendering())	Property:SeriesRender @(Html.EJS().AccumulationChart("container").SeriesRender())
Fire s before	Property:TitleRendering @(Html.EJ().Chart("chartContainer").TitleRendering())	Not applicable

rendering the Chart title.		
Fires before rendering the Chart subtitle.	<code>Property:SubTitleRendering@(Html.EJ().Chart("chartContainer").SubTitleRendering())</code>	Not applicable
Fires before rendering the tooltip.	<code>Property:TooltipInitialize@(Html.EJ().Chart("chartContainer").TooltipInitialize())</code>	<code>Property:TooltipRender@(Html.EJS().AccumulationChart("container").TooltipRender())</code>

AppBar

Getting Started with ASP.NET MVC AppBar Control

This section briefly explains about how to include ASP.NET MVC AppBar control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC AppBar control

Now, add the Syncfusion ASP.NET MVC AppBar control in ~/Home/Index.cshtml page.

CSHTML

```
<div class="col-lg-12 control-section default-appbar-section">
@ (Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentT
emplate(@<div>
@Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-
icons e-menu").Render()
<span class="regular" style="margin:0 5px">EJ2 AppBar</span>
<div class="e-appbar-spacer"></div>
@Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("FREE
TRIAL").Render()
</div>).Render())
</div>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC AppBar control will be rendered in the default web browser.



Note: You can refer to our [ASP.NET MVC AppBar](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC AppBar Example](#) that shows you how to render and configure the AppBar in ASP.NET MVC.

Size and Color with ASP.NET MVC AppBar Control

Size

The size of the AppBar can be set using the mode property. The available types of the AppBar are as follows:

- Regular AppBar
- Prominent AppBar
- Dense AppBar

Regular AppBar

This mode is the default one in which the AppBar is displayed with the default height.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
```

```
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentTemplate(@<div>
    @(Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
    <span class="regular">Regular AppBar</span>
    <div class="e-appbar-spacer"></div>
    @(Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("FREE TRIAL").Render()
</div>).Render())
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin:0 3px;
    }
</style>
```

REGULAR.CS

```
public ActionResult Index()
{
    return View();
}
```



Prominent AppBar

This height mode can be set to the AppBar by setting **Prominent** to the mode property. The prominent AppBar is displayed with a longer height and can be used for larger titles, images, or texts. It is also longer than the regular AppBar. In the following example, we have customized the prominent text using align-self and white-space CSS properties. You can change the prominent AppBar height if larger titles, images, or texts are used.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).Mode(AppBarMode.Prominent).CssClass("prominent-appbar").ContentTemplate(@<div>
    @(Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
    <span class="prominent">AppBar Component with Prominent mode</span>
    <div class="e-appbar-spacer"></div>
    @(Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("FREE TRIAL").Render()
</div>).Render())
<style>
    .prominent-appbar .prominent {
        align-self: center;
        white-space: break-spaces;
        text-align: inherit;
        font-size: 35px;
        line-height: 50px;
    }
</style>
```

```

.prominent-appbar.e-appbar {
    background-image:
url("https://ej2.syncfusion.com/demos/src/appbar/images/prominent.png");
    background-size: 100% 400px;
    color: #ffffff;
    background-repeat: no-repeat;
    height: 400px;
}
.prominent-appbar .e-inherit.e-btn {
    background: transparent;
}
.prominent-appbar .e-inherit.e-btn:hover,
.prominent-appbar .e-inherit.e-btn:focus,
.prominent-appbar .e-inherit.e-btn:active,
.prominent-appbar .e-inherit.e-btn.e-active,
.prominent-appbar .e-inherit.e-css.e-btn:hover,
.prominent-appbar .e-inherit.e-css.e-btn:focus
.prominent-appbar .e-inherit.e-css.e-btn:active
.prominent-appbar .e-inherit.e-css.e-btn.e-active {
    background: rgba(255, 255, 255, .08);
}
</style>

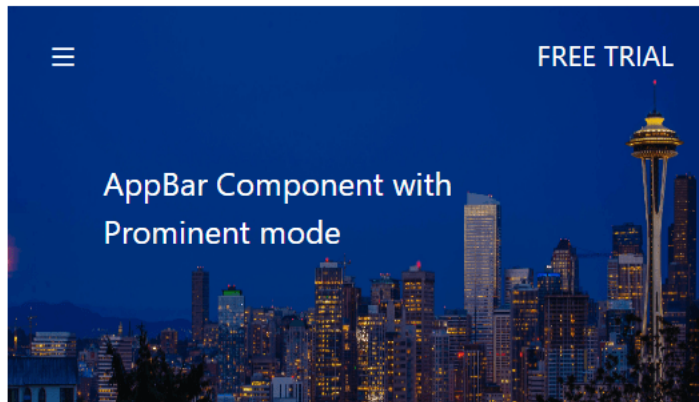
```

PROMINENT.CS

```

public ActionResult Index()
{
    return View();
}

```



Dense AppBar

This height mode can be set to the AppBar by setting **Dense** to the mode property. Dense AppBar is displayed with shorter height which is denser to accommodate all the AppBar content.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).Mode(AppBarMode.Dense).ContentTemplate(@<div>

```

```

@Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
<span class="dense">Dense AppBar</span>
<div class="e-appbar-spacer"></div>
@Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("FREE TRIAL").Render()
</div>).Render()
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin: 0 3px;
    }
</style>


```

DENSE.CS

```

public ActionResult Index()
{
    return View();
}

```



Color

The background and font colors can be set using the color mode property. The available types of background color for the AppBar are as follows:

- Light AppBar
- Dark AppBar
- Primary AppBar
- Inherit AppBar

Light AppBar

This color mode is the default one in which the AppBar can be displayed with a light background and its corresponding font color.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().AppBar("defaultAppBar").ContentTemplate(@<div>
    <a href="https://www.syncfusion.com/aspnet-mvc-ui-controls"
    target="_blank" rel="noopener" role="link" aria-label="Syncfusion MVC
    controls">
        <div class="syncfusion-logo"></div>
    </a>
    <div class="e-appbar-spacer"></div>
    @Html.EJS().Button("defaultButtonLogin").IsPrimary(true).Content("FREE
    TRIAL").Render()
</div>).Render()
<style>
    #defaultAppBar .syncfusion-logo {

```

```

        background:
url (https://cdn.syncfusion.com/blazor/images/demos/syncfusion-logo.svg);
        background-size: contain;
        background-repeat: no-repeat;
        height: 30px;
        width: 150px;
    }
</style>

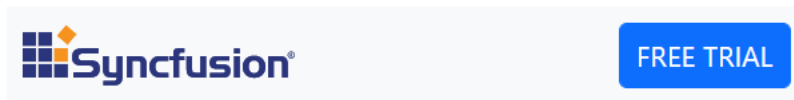
```

LIGHT.CS

```

public ActionResult Index()
{
    return View();
}

```

*Dark AppBar*

This color mode can be set to the AppBar by setting **Dark** to the property color mode. A dark AppBar can be displayed with a dark background and its corresponding font color.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Dark).ContentTemplate(
    @<div>
        @Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
        <div class="e-appbar-spacer"></div>
        @Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("FREE TRIAL").Render()
    </div>
    .Render())
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin:0 3px;
    }
</style>

```

DARK.CS

```

public ActionResult Index()
{
    return View();
}

```



Primary AppBar

This color mode can be set to the AppBar by setting **Primary** to the property color mode. The primary AppBar can be displayed with primary colors.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS() .AppBar ("defaultAppBar") .ColorMode (AppBarColor.Primary) .ContentT
emplate (@<div>
    @Html.EJS() .Button ("defaultButtonMenu") .CssClass ("e-
inherit") .IconCss ("e-icons e-menu") .Render ()
    <div class="e-appbar-spacer"></div>
    @Html.EJS() .Button ("defaultButtonLogin") .CssClass ("e-
inherit") .Content ("FREE TRIAL") .Render ()
</div>) .Render () )
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin:0 3px;
    }
</style>
```

PRIMARY.CS

```
public ActionResult Index()
{
    return View();
}
```



Inherit AppBar

This color mode can be set to the AppBar by setting **Inherit** to the property color mode. The AppBar inherits the background and font color from its parent element.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS() .AppBar ("defaultAppBar") .ColorMode (AppBarColor.Inherit) .ContentT
emplate (@<div>
    <a href="https://www.syncfusion.com/aspnet-mvc-ui-controls"
target="_blank" rel="noopener" role="link" aria-label="Syncfusion MVC
controls">
        <div class="syncfusion-logo"></div>
    </a>
    <div class="e-appbar-spacer"></div>
    @Html.EJS() .Button ("defaultButtonLogin") .IsPrimary (true) .Content ("FREE
TRIAL") .Render ()
</div>) .Render () )
<style>
    #defaultAppBar .syncfusion-logo {
```

```

        background:
url (https://cdn.syncfusion.com/blazor/images/demos/syncfusion-logo.svg) ;
        background-size: contain;
        background-repeat: no-repeat;
        height: 30px;
        width: 150px;
    }
</style>

```

INHERIT.CS

```

public ActionResult Index()
{
    return View();
}

```



Accessibility in ASP.NET MVC AppBar Control

The AppBar control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the AppBar control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |


```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>
```

Keyboard interaction

The AppBar control provides the focus element navigation based on its's tab key order.

Ensuring accessibility

The AppBar control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the AppBar control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the AppBar control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Positioning in ASP.NET MVC AppBar Control

The position of the AppBar can be set using the position and sticky property. The AppBar provides the following options for setting its position:

- Top AppBar
- Bottom AppBar
- Sticky AppBar

Top AppBar

The top AppBar is the default one in which it positions the AppBar at the top of the content.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
<div class="default-appbar-section">

@ (Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentT
emplate(@<div>
    @Html.EJS().Button("defaultButtonMenu").CssClass("e-
inherit").IconCss("e-icons e-menu").Render())
```

```

        <div class="e-appbar-spacer"></div>
        @Html.EJS().Button("defaultButtonLogin").CssClass("e-
inherit").Content("FREE TRIAL").Render()
    </div>).Render()
    <div class="appbar-content" style="font-size: 12px">
        <p>
            The AppBar also known as action bar or nav bar displays
            information and actions related to the current application screen. It is
            used to show branding, screen titles, navigation, and actions. The control
            supports height mode, color mode, positioning, and more.
        </p>
        <p>
            The AppBar control provides flexible ways to configure the look
            and feel of the bar to match your requirement.
        </p>
        <p>
            Developers can control the appearance and behaviors of the
            AppBar using a rich set of APIs.
        </p>
        <p>
            The AppBar component supports built-in themes such as Material,
            Bootstrap, Fabric (Office 365), Tailwind CSS, and high contrast. Users can
            customize these built-in themes or create new themes to achieve their
            desired look and feel by either simply overriding SASS variables or using
            our Theme Studio application.
        </p>
    </div>
</div>
<style>
    .default-appbar-section {
        height: 220px;
        margin: 0 auto;
        width: 500px;
        overflow-y: scroll;
    }
    .default-appbar-section .e-btn.e-inherit {
        margin: 0 3px;
    }
</style>

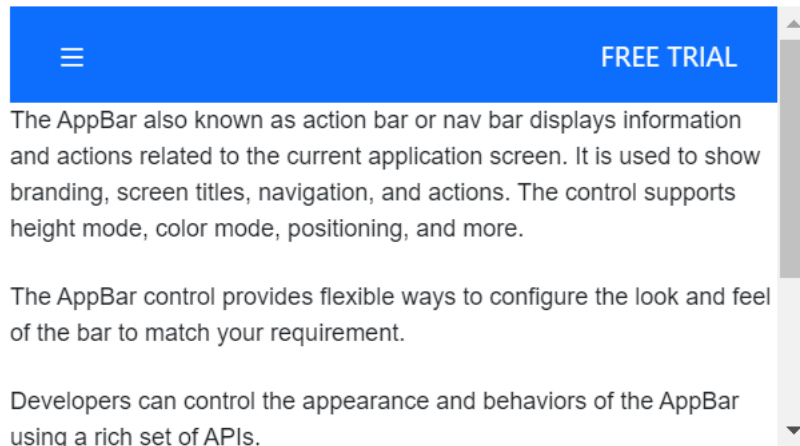
```

TOP.CS

```

public ActionResult Index()
{
    return View();
}

```



Bottom AppBar

This position can be set to the AppBar by setting **Bottom** to the position property. The bottom AppBar positions the AppBar at the bottom of the content.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
<div class="default-appbar-section">

@ (Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).Position
(AppBarPosition.Bottom).ContentTemplate(@<div>
    @Html.EJS().Button("defaultButtonMenu").CssClass("e-
inherit").IconCss("e-icons e-menu").Render()
    <div class="e-appbar-spacer"></div>
    @Html.EJS().Button("defaultButtonLogin").CssClass("e-
inherit").Content("FREE TRIAL").Render()
    </div>).Render()
    <div class="appbar-content" style="font-size: 12px">
    <p>
        The AppBar also known as action bar or nav bar displays
        information and actions related to the current application screen. It is
        used to show branding, screen titles, navigation, and actions. The control
        supports height mode, color mode, positioning, and more.
    </p>
    <p>
        The AppBar control provides flexible ways to configure the look
        and feel of the bar to match your requirement.
    </p>
    <p>
        Developers can control the appearance and behaviors of the
        AppBar using a rich set of APIs.
    </p>
    <p>
        The AppBar component supports built-in themes such as Material,
        Bootstrap, Fabric (Office 365), Tailwind CSS, and high contrast. Users can
        customize these built-in themes or create new themes to achieve their
        desired look and feel by either simply overriding SASS variables or using
        our Theme Studio application.
    </p>
    </div>
    </div>
```

```

</div>
</div>
<style>
    .default-appbar-section {
        height: 420px;
        width: 500px;
        margin: 0 auto;
        position: relative;
    }
    .default-appbar-section .e-btn.e-inherit {
        margin: 0 3px;
    }
</style>

```

BOTTOM.CS

```

public ActionResult Index()
{
    return View();
}

```

The AppBar also known as action bar or nav bar displays information and actions related to the current application screen. It is used to show branding, screen titles, navigation, and actions. The control supports height mode, color mode, positioning, and more.

The AppBar control provides flexible ways to configure the look and feel of the bar to match your requirement.

Developers can control the appearance and behaviors of the AppBar using a rich set of APIs.

The AppBar component supports built-in themes such as Material, Bootstrap, Fabric (Office 365), Tailwind CSS, and high contrast. Users can customize these built-in themes or create new themes to achieve their desired look and feel by either simply overriding SASS variables or using our Theme Studio application.



Sticky AppBar

This position can be set to the AppBar by setting `true` to the sticky property. AppBar will be sticky while scrolling the AppBar content.

CSHTML

```

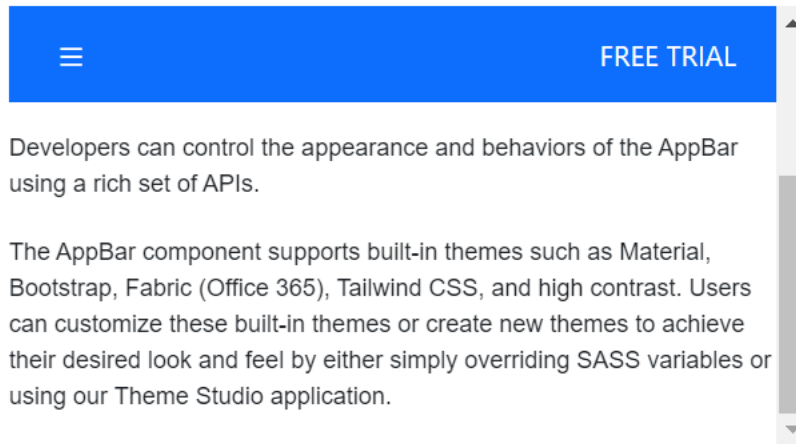
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).IsSticky(
true).ContentTemplate(@<div>

```

```
@Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
<div class="e-appbar-spacer"></div>
@Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("FREE TRIAL").Render()
</div>).Render()
<div class="appbar-content" style="font-size: 12px">
<p>
    The AppBar also known as action bar or nav bar displays information and actions related to the current application screen. It is used to show branding, screen titles, navigation, and actions. The control supports height mode, color mode, positioning, and more.
</p>
<p>
    The AppBar control provides flexible ways to configure the look and feel of the bar to match your requirement.
</p>
<p>
    Developers can control the appearance and behaviors of the AppBar using a rich set of APIs.
</p>
<p>
    The AppBar component supports built-in themes such as Material, Bootstrap, Fabric (Office 365), Tailwind CSS, and high contrast. Users can customize these built-in themes or create new themes to achieve their desired look and feel by either simply overriding SASS variables or using our Theme Studio application.
</p>
</div>
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin:0 3px;
    }
</style>
```

STICKY.CS

```
public ActionResult Index()
{
    return View();
}
```



Design User Interface with ASP.NET MVC AppBar Control

Spacer

Spacer is used to provide spacing between the AppBar contents which gives additional space to the content layout.

The following example depicts the code to provide spacing between the home and pan buttons in the AppBar:

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentTemplate(@<div>
    @Html.EJS().Button("defaultButtonHome").CssClass("e-inherit").IconCss("e-icons e-home").Render()
    <div class="e-appbar-spacer"></div>
    @Html.EJS().Button("defaultButtonCut").CssClass("e-inherit").IconCss("e-icons e-cut").Render()
    <div class="e-appbar-spacer"></div>
    @Html.EJS().Button("defaultButtonPan").CssClass("e-inherit").IconCss("e-icons e-pan").Render()
</div>).Render())
```

SPACER.CS

```
public ActionResult Index()
{
    return View();
}
```



Separator

Separator shows a vertical line to visually group or separate the AppBar contents.

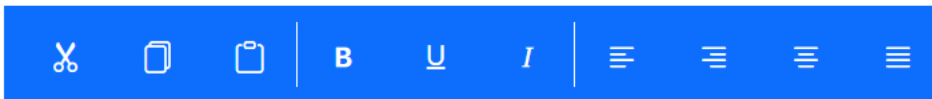
The following example depicts the code to provide a vertical line between a group of buttons in the AppBar.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentTemplate(@<div>
    @Html.EJS().Button("defaultButtonCut").CssClass("e-inherit").IconCss("e-icons e-cut").Render()
    @Html.EJS().Button("defaultButtonCopy").CssClass("e-inherit").IconCss("e-icons e-copy").Render()
    @Html.EJS().Button("defaultButtonPaste").CssClass("e-inherit").IconCss("e-icons e-paste").Render()
    <div class="e-appbar-separator"></div>
    @Html.EJS().Button("defaultButtonBold").CssClass("e-inherit").IconCss("e-icons e-bold").Render()
    @Html.EJS().Button("defaultButtonUnderline").CssClass("e-inherit").IconCss("e-icons e-underline").Render()
    @Html.EJS().Button("defaultButtonItalic").CssClass("e-inherit").IconCss("e-icons e-italic").Render()
    <div class="e-appbar-separator"></div>
    @Html.EJS().Button("defaultButtonAlignLeft").CssClass("e-inherit").IconCss("e-icons e-align-left").Render()
    @Html.EJS().Button("defaultButtonAlignRight").CssClass("e-inherit").IconCss("e-icons e-align-right").Render()
    @Html.EJS().Button("defaultButtonAlignCenter").CssClass("e-inherit").IconCss("e-icons e-align-center").Render()
    @Html.EJS().Button("defaultButtonJustify").CssClass("e-inherit").IconCss("e-icons e-justify").Render()
</div>).Render())
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin: 0 3px;
    }
</style>
```

SEPARATOR.CS

```
public ActionResult Index()
{
    return View();
}
```



Media Query

Media Query is used to adjusting the AppBar for different screen sizes. Resize the screen to observe the responsive layout of the AppBar.

CSHTML

```
@using Syncfusion.EJ2;
```

```

@using Syncfusion.EJ2.Navigations;
<div class="default-appbar-section">

@ (Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentT
emplate(@<div>
    @Html.EJS().Button("defaultButtonMenu").CssClass("e-
inherit").IconCss("e-icons e-menu").Render()
    @Html.EJS().Button("defaultButtonHome").CssClass("e-
inherit").Content("Home").Render()
    @Html.EJS().Button("defaultButtonAbout").CssClass("e-
inherit").Content("About").Render()
    @Html.EJS().Button("defaultButtonProducts").CssClass("e-
inherit").Content("Products").Render()
    @Html.EJS().Button("defaultButtonContacts").CssClass("e-
inherit").Content("Contacts").Render()
    <div class="e-appbar-spacer"></div>
    <div class="e-appbar-separator"></div>
    @Html.EJS().Button("defaultButtonLogin").CssClass("e-
inherit").Content("Login").Render()
    </div>).Render()
</div>
<style>
    .default-appbar-section .e-btn.e-inherit {
        margin: 0 3px;
    }
    @@media screen and (max-width: 1024px) {
        .default-appbar-section .e-appbar,
        .default-appbar-section .e-appbar > div:first-child {
            flex-flow: row wrap;
            height: auto;
            gap: 8px;
        }
        .default-appbar-section {
            width: 350px;
        }
    }
    @@media screen and (max-width: 480px) {
        .default-appbar-section {
            width: 200px;
            margin: 0 2px;
        }
    }
</style>

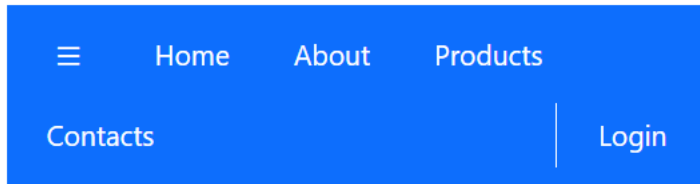
```

MEDIA.CS

```

public ActionResult Index()
{
    return View();
}

```

Designing AppBar with Menu

AppBar is rendered with a Menu component in its AppBar header area. Menu component's styles are inherited from the AppBar component using the `e-inherit` CSS class.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentTemplate(
    @(<div>
        @Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
        @Html.EJS().Menu("defaultMenuCompany").CssClass("e-inherit").Items(ViewBag.CompanyMenuItems).Render()
        @Html.EJS().Menu("defaultMenuProducts").CssClass("e-inherit").Items(ViewBag.ProductMenuItems).Render()
        @Html.EJS().Menu("defaultMenuAbout").CssClass("e-inherit").Items(ViewBag.AboutMenuItems).Render()
        @Html.EJS().Menu("defaultMenuCarrers").CssClass("e-inherit").Items(ViewBag.CarrerMenuItems).Render()
        <div class="e-appbar-spacer"></div>
        @Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("Login").Render()
    </div>).Render())
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin: 0 3px;
    }
</style>
```

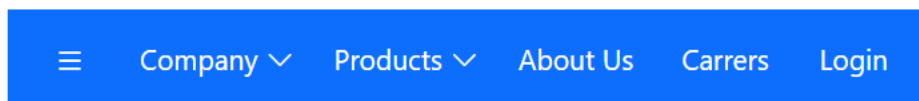
MENU.CS

```
using Syncfusion.EJ2.Navigations;
public ActionResult Index()
{
    List<MenuItem> companyMenuItems = new List<MenuItem>() {
        new MenuItem
        {
            Text = "Company",
            Items = new List<MenuItem>()
            {
                new MenuItem { Text= "About Us" },
                new MenuItem { Text= "Customers" },
                new MenuItem { Text= "Blog" },
                new MenuItem { Text= "Careers" }
            }
        }
    };
    ViewBag.CompanyMenuItems = companyMenuItems;
```

```

List<MenuItem> productMenuItems = new List<MenuItem>() {
    new MenuItem
    {
        Text = "Products",
        Items = new List<MenuItem>() {
            new MenuItem { Text= "Developer" },
            new MenuItem { Text= "Analytics" },
            new MenuItem { Text= "Reporting" },
            new MenuItem { Text= "Help Desk" }
        }
    }
};
ViewBag.ProductMenuItems = productMenuItems;
List<MenuItem> aboutMenuItems = new List<MenuItem>() {
    new MenuItem
    {
        Text = "About Us"
    }
};
ViewBag.AboutMenuItems = aboutMenuItems;
List<MenuItem> carrerMenuItems = new List<MenuItem>() {
    new MenuItem
    {
        Text = "Carrers"
    }
};
ViewBag.CarrerMenuItems = carrerMenuItems;
return View();
}

```



Designing AppBar with Buttons

The AppBar is rendered with a Button and DropDownButton component in its AppBar header area. Button and DropDownButton components' styles are inherited from the AppBar component using the `e-inherit` CSS class.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentTemplate(
    @<div>
        @Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-menu").Render()
        @Html.EJS().DropDownButton("defaultDropDownButtonProduct").CssClass("e-inherit").Content("Products").Items(ViewBag.ProductDropDownButtonItem
        er()
        <div class="e-appbar-spacer"></div>
        @Html.EJS().Button("defaultButtonLogin").CssClass("e-inherit").Content("Login").Render()
    </div>).Render()
)

```

```
<style>
    #defaultAppBar .e-btn.e-inherit {
        margin: 0 3px;
    }
</style>
```

BUTTON.CS

```
public ActionResult Index()
{
    List<object> productDropDownButtonItem = new List<object>();
    productDropDownButtonItem.Add(new
    {
        text = "Developer",
    });
    productDropDownButtonItem.Add(new
    {
        text = "Analytics",
    });
    productDropDownButtonItem.Add(new
    {
        text = "Reporting",
    });
    productDropDownButtonItem.Add(new
    {
        text = "E-Signature",
    });
    productDropDownButtonItem.Add(new
    {
        text = "Help-Desk",
    });
    ViewBag.ProductDropDownButtonItem = productDropDownButtonItem;
    return View();
}
```



Designing AppBar with SideBar

The AppBar is rendered with the SideBar component below the AppBar. Click on the menu icon to expand/collapse the Sidebar. In the following sample, the `toggle` method has been used to show or hide the Sidebar on the AppBar button click.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
<div class="default-appbar-section">
    <div>
        @ (Html.EJS().AppBar("defaultAppBar").ContentTemplate(@<div>
            @Html.EJS().Button("defaultButtonMenu").CssClass("e-
inherit").IconCss("e-icons e-menu").Click("toggle").Render()
            <div class="e-folder">
                <div class="e-folder-name">Navigation Pane</div>
```

```

        </div>
    </div>).Render()
</div>
@{Html.EJS().Sidebar("sideTree").Width("290px").Target(".main-
content").MediaQuery("(min-width:
600px)").IsOpen(true).ContentTemplate(@<div>
    <div class='main-menu'>
        <div class="table-content">

@Html.EJS().TextBox("resSearch").Placeholder("Search...").Render()
        <p class="main-menu-header">TABLE OF CONTENTS</p>
    </div>
    <div>
        @Html.EJS().TreeView("mainTree").CssClass("main-
treeview").ExpandOn(Syncfusion.EJ2.Navigations.ExpandOnSettings.Click).Field
s(new Syncfusion.EJ2.Navigations.TreeViewFieldsSettings() { Id = "nodeId",
Text = "nodeText", DataSource = ViewBag.TreeData, HasChildren = "hasChild",
ParentID = "pid" }).Render()
    </div>
</div>
</div>).HtmlAttributes(ViewBag.HtmlAttribute).Render();}
<div class="main-content" id="main-text">
    <div class="sidebar-content">
        <div class="sidebar-heading"> Responsive Sidebar with
TreeView</div>
        <p class="paragraph-content">
            This is a graphical aid for visualising and categorising the
            site, in the style of an expandable and collapsable treeview component.
            It auto-expands to display the node(s), if any,
            corresponding to the currently viewed title, highlighting that node(s)
            and its ancestors. Load-on-demand when expanding nodes is
            available where supported (most graphical browsers),
            falling back to a full-page reload. MediaWiki-supported
            caching, aside from squid, has been considered so that
            unnecessary re-downloads of content are avoided where
            possible. The complete expanded/collapsed state of
            the treeview persists across page views in most situations.
        </p>
    </div>
</div>
</div>
<script>
document.addEventListener('DOMContentLoaded', function () {
    sidebarObj = document.getElementById("sideTree").ej2_instances[0];
});
function toggle() {
    sidebarObj.toggle();
}
</script>
<style>
    .default-appbar-section .e-appbar .e-folder {
        margin:0 5px;
    }
    .container-fluid {
        padding-left: 0;
        padding-right: 0;
    }

```

```
.sidebar-treeview .e-treeview .e-icon-collapsible,
.sidebar-treeview .e-treeview .e-icon-expandable {
    float: right;
}
.sidebar-treeview .e-treeview .e-icon-collapsible,
.sidebar-treeview .e-treeview .e-icon-expandable {
    margin: 3px;
}
.sidebar-treeview .e-treeview,
.sidebar-treeview .e-treeview .e-ul {
    padding: 0;
    margin: 0;
}
.default-appbar-section .sidebar-treeview {
    z-index: 20 !important;
}
.sidebar-treeview .main-menu .main-menu-header {
    color: #656a70;
    padding: 15px 15px 15px 0;
    font-size: 14px;
    width: 13em;
    margin: 0;
}
#main-text .sidebar-heading {
    font-size: 16px;
}
.sidebar-treeview .table-content {
    padding: 20px 18px;
    height: 8em;
}
#main-text .sidebar-content .line {
    width: 100%;
    height: 1px;
    border-bottom: 1px dashed #ddd;
    margin: 40px 0;
}
#main-text .sidebar-content {
    padding: 15px;
    font-size: 14px;
}
#main-text .paragraph-content {
    padding: 15px 0;
    font-weight: normal;
    font-size: 14px;
}
.sidebar-treeview .main-treeview .icon {
    font-family: 'fontello';
    font-size: 16px;
    margin: -4px 0;
}
.e-folder {
    text-align: center;
    font-weight: 500;
    font-size: 16px;
}
.sidebar-treeview .e-treeview .e-text-content {
    padding-left: 18px;
```

```

    }
    #wrapper .e-toolbar {
        border-bottom: 1px solid #d2d6de;
    }
    .default-appbar-section .main-content {
        height: 380px;
    }
    .sidebar-treeview {
        border-right: 1px solid #dee2e6 !important;
    }
    .e-folder-name {
        margin-top: -2px;
    }
}
</style>

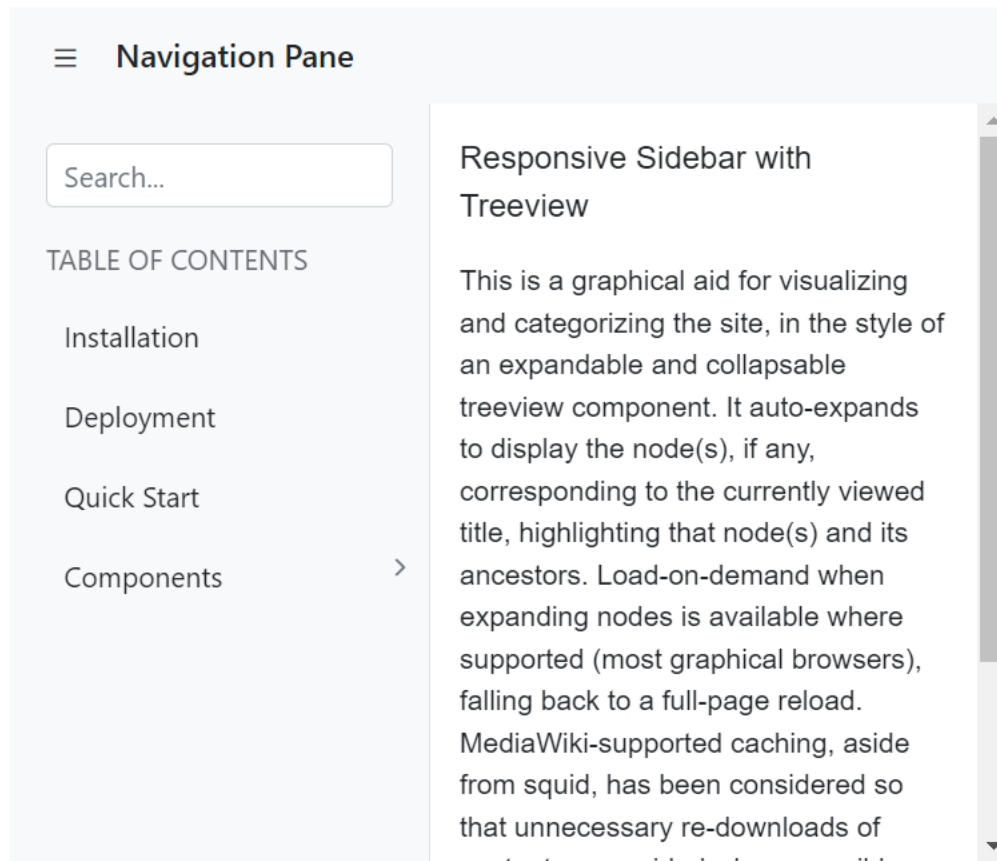
```

SIDEBAR.CS

```

public ActionResult Index()
{
    Dictionary<string, object> htmlAttribute = new Dictionary<string,
object>()
    {
        {"class", "sidebar-treeview" } };
    ViewBag.HtmlAttribute = htmlAttribute;
    List<TreeDatas> treedata = new List<TreeDatas>();
    treedata.Add(new TreeDatas { nodeId = "01", nodeText = "Installation"
});
    treedata.Add(new TreeDatas { nodeId = "02", nodeText = "Deployment" });
    treedata.Add(new TreeDatas { nodeId = "03", nodeText = "Quick Start" });
    treedata.Add(new TreeDatas { nodeId = "04", nodeText = "Components",
hasChild = true });
    treedata.Add(new TreeDatas { nodeId = "04-01", nodeText = "Calendar",
pid = "04" });
    treedata.Add(new TreeDatas { nodeId = "04-02", nodeText = "DatePicker",
pid = "04" });
    treedata.Add(new TreeDatas { nodeId = "04-03", nodeText =
"DateTimePicker", pid = "04" });
    treedata.Add(new TreeDatas { nodeId = "04-04", nodeText =
"DateRangePicker", pid = "04" });
    treedata.Add(new TreeDatas { nodeId = "04-05", nodeText = "TimePicker",
pid = "04" });
    treedata.Add(new TreeDatas { nodeId = "04-06", nodeText = "SideBar", pid
= "04" });
    ViewBag.TreeData = treedata;
    return View();
}
public class TreeDatas
{
    public string nodeId { get; set; }
    public string nodeText { get; set; }
    public bool hasChild { get; set; }
    public string pid { get; set; }
}

```



Styles and Appearances in ASP.NET MVC AppBar Control

To modify the AppBar appearance, you need to override the default CSS of the AppBar component. Find the list of CSS classes and their corresponding sections in the AppBar component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class	Purpose of Class
-----	-----
.e-appbar	To customize the appbar.
.e-appbar.e-prominent	To customize the prominent appbar.
.e-appbar.e-dense	To customize the dense appbar.
.e-appbar.e-light	To customize the light appbar.
.e-appbar.e-dark	To customize the dark appbar.
.e-appbar.e-primary	To customize the dark appbar.
.e-appbar.e-inherit	To customize the inherit appbar.

Note: You can change the prominent AppBar height if larger titles, images, or texts are used.

CssClass

CssClass is used for AppBar customization based on the custom class. In the example below, the AppBar background and color are customized using the CssClass property.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).CssClass("custom-appbar").ContentTemplate(@<div>
    @Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-home").Render()
</div>).Render())
<style>
    .e-appbar.custom-appbar {
        background: #ff0000;
        color: #fff;
    }
</style>
```

CSSCLASS.CS

```
public ActionResult Index()
{
    return View();
}
```

**HtmlAttributes**

It can be used for additional inline attributes by specifying as inline attributes or by specifying `htmlAttributes` directive. In the code example below, the `aria-label` of the AppBar is customized by specifying as attributes.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().AppBar("defaultAppBar").ColorMode(AppBarColor.Primary).ContentTemplate(@<div>
    @Html.EJS().Button("defaultButtonMenu").CssClass("e-inherit").IconCss("e-icons e-home").Render()
</div>).HtmlAttributes(ViewBag.HtmlAttribute).Render())
```

HTMLATTRIBUTES.CS

```
public ActionResult Index()
{
    Dictionary<string, object> htmlAttribute = new Dictionary<string, object>()
    {
        { "aria-label", "appbar" } };
    ViewBag.HtmlAttribute = htmlAttribute;
    return View();
}
```


Auto Complete

Getting Started with ASP.NET MVC AutoComplete Control

This section briefly explains about how to include [ASP.NET MVC AutoComplete](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{ site.ej2version
}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC AutoComplete control

Now, add the Syncfusion ASP.NET MVC AutoComplete control in `~/Home/Index.cshtml` page.

CSHTML

```
@model List<string>
@Html.EJS().AutoComplete("games").DataSource(Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC AutoComplete control will be rendered in the default web browser.



Note: Running the above code will display the basic AutoComplete on the browser.

Binding data source

After initialization, populate the AutoComplete with data using the [dataSource](#) property. Here, an array of string values is passed to the AutoComplete control.

CSHTML

```
@model List<string>
@Html.EJS().AutoComplete("games").Placeholder("Select a
game").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Custom values

The AutoComplete allows the user to give input as custom value which is not required to present in predefined set of values. By default, this support is enabled by [allowCustom](#) property. The custom value will be sent to post back handler when a form is about to be submitted.

CSHTML

```
@model List<string>
@Html.EJS().AutoComplete("games").AllowCustom(true).Placeholder("Select a
game").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Configure the suggestion list

By default, suggestion list width automatically adjusts according to the AutoComplete input element's width, and the height of the suggestion list has '300px'.

The height and width of the popup list can also be customized using the [popupHeight](#) and [popupWidth](#) property respectively.

In the following sample, suggestion list's width and height are configured.

CSHTML

```
@model List<string>
```

```
@Html.EJS().AutoComplete("games").Placeholder("Select a
game").PopupWidth("300px").PopupHeight("200px").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Note: [View Sample in GitHub.](#)

See also

- [How to bind the data](#)

Data binding in ASP.NET MVC AutoComplete Control

The AutoComplete loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of array or [DataManager](#).

The AutoComplete also supports different kind of data services such as OData, OData V4, Web API and data formats such as XML, JSON, JSONP with the help of DataManager Adaptors.

| Fields | Type | Description |

|-----|-----|-----|

| value | **number or string** | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| groupBy | **string** | Specifies the category under which the list item has to be grouped. |

| iconCss | **string** | Specifies the icon class of each list item. |

Note: While binding complex data to AutoComplete, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Bind to local data

Local data can be represented in two ways as described below.

Array of string

The AutoComplete has support to load array of primitive data such as strings and numbers.

CSHTML

```
@Html.EJS().AutoComplete("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Render()
```

ARRAYOFSTRINGS.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult arrayofstring()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball",
            "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}

```

Array of object

The AutoComplete can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Name** column from complex data have been mapped to the **value** field.

CSHTML

```

@Html.EJS().AutoComplete("country").Placeholder("Select a
Country").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data)
.Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value =
"Name" }).Render()

```

ARRAYOFOBJECTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult arrayofobjects()
        {
            ViewBag.data = new Countries().CountriesList();
            return View();
        }
    }
}

```

Array of complex object

The AutoComplete can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Country.CountryId` column from complex data have been mapped to the `value` field.

CSHTML

```
@Html.EJS().AutoComplete("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<Object>)ViewBag.data).Fields(
new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "Country.CountryId" }).Render()
```

COMPLEX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Code
    {
        public string Id { get; set; }
    }
    public class Country
    {
        public string CountryId { get; set; }
    }
    public class Complex
    {
        public Country Country { get; set; }
        public Code Code { get; set; }
        public List<Complex> GetData()
        {
            List<Complex> data = new List<Complex>();
            data.Add(new Complex() { Country = new Country() { CountryId =
"Australia" }, Code = new Code() { Id = "AU" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Bermuda" }, Code = new Code() { Id = "BM" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Canada" }, Code = new Code() { Id = "CA" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Cameroon" }, Code = new Code() { Id = "CM" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Denmark" }, Code = new Code() { Id = "DK" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"France" }, Code = new Code() { Id = "FR" } });
            return data;
        }
    }
}
```

Bind to remote data

The AutoComplete supports retrieval of data from remote data services with the help of [DataManager](#) control. The [Query](#) property is used to fetch data from the database and bind it to the AutoComplete.

The following sample displays the first 6 contacts from the **Customers** table of the **Northwind** data service.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "ContactName"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
```

REMOTEDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult remotedata()
        {
            return View();
        }
    }
}
```

Bind to URL Adaptor

The AutoComplete supports retrieval of data from URL adaptor.

CSHTML

```
@Html.EJS().AutoComplete("games").Placeholder("Select a Country").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("/AutoComplete/UrlDatasource").Adaptor("UrlAdaptor").CrossDomain(true)).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "ShipCountry"
}).Render()
```

URLDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Threading.Tasks;
using System.Web.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult UrlDatasource([FromBody]Data dm)
        {
            var val = OrdersDetails.GetAllRecords();
            var Data = val.ToList();
            var count = val.Count();
            if (dm.where != null)
            {
                Data = (from cust in Data
                        where
cust.ShipCountry.ToLower().StartsWith(dm.@where[0].value.ToString())
                        select cust).ToList();
            }
            if (dm.take != 0)
                Data = Data.Take(dm.take).ToList();
            return Json(Data);
        }
        public class Data
        {
            public int take { get; set; }
            public List<Wheres> where { get; set; }
        }
        public class Wheres
        {
            public string field { get; set; }
            public bool ignoreAccent { get; set; }
            public bool ignoreCase { get; set; }
            public bool isComplex { get; set; }
            public string value { get; set; }
            public string Operator { get; set; }
        }
        public class OrdersDetails
        {
            public static List<OrdersDetails> order = new
List<OrdersDetails>();
            public OrdersDetails()
            {
            }
            public OrdersDetails(int OrderID, string CustomerId, int
EmployeeId, double Freight, bool Verified, DateTime OrderDate, string
ShipCity, string ShipName, string ShipCountry, DateTime ShippedDate, string
ShipAddress)
            {
                this.OrderID = OrderID;
                this.CustomerID = CustomerId;
                this.EmployeeID = EmployeeId;
                this.Freight = Freight;
            }
        }
    }
}

```



```

        this.ShipCity = ShipCity;
        this.Verified = Verified;
        this.OrderDate = OrderDate;
        this.ShipName = ShipName;
        this.ShipCountry = ShipCountry;
        this.ShippedDate = ShippedDate;
        this.ShipAddress = ShipAddress;
    }
    public static List<OrdersDetails> GetAllRecords()
    {
        if (order.Count() == 0)
        {
            int code = 10000;
            for (int i = 1; i < 10; i++)
            {
                order.Add(new OrdersDetails(code + 1, "ALFKI", i +
0, 2.3 * i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro",
"Denmark", new DateTime(1996, 7, 16), "Kirchgasse 6"));
                order.Add(new OrdersDetails(code + 2, "ANATR", i +
2, 3.3 * i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha",
"Brazil", new DateTime(1996, 9, 11), "Avda. Azteca 123"));
                order.Add(new OrdersDetails(code + 3, "ANTON", i +
1, 4.3 * i, true, new DateTime(1957, 11, 30), "Cholchester",
"Frankenversand", "Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave.
Bolívar #65-98 Llano Largo"));
                order.Add(new OrdersDetails(code + 4, "BLONP", i +
3, 5.3 * i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel",
"Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
                order.Add(new OrdersDetails(code + 5, "BOLID", i +
4, 6.3 * i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes",
"Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
                code += 5;
            }
        }
        return order;
    }
    public int? OrderID { get; set; }
    public string CustomerID { get; set; }
    public int? EmployeeID { get; set; }
    public double? Freight { get; set; }
    public string ShipCity { get; set; }
    public bool Verified { get; set; }
    public DateTime OrderDate { get; set; }
    public string ShipName { get; set; }
    public string ShipCountry { get; set; }
    public DateTime ShippedDate { get; set; }
    public string ShipAddress { get; set; }
}
}
}

```

Web API Adaptor

Use the **WebApiAdaptor** to bind autocomplete with Web API created using OData.

CSHTML

```
<div id='web-data' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().AutoComplete("games").DataSource(dataManger =>
dataManger.Url("/api/AutoComplete/").Adaptor("WebApiAdaptor").Render()
    </div>
</div>
```

WEBAPI.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    [Route("api/[controller]")]
    public class AutoCompleteController : Controller
    {
        List<string> game = new List<string>();
        [HttpGet]
        public List<string> Get()
        {
            game.Add("Badminton");
            game.Add("Basketball");
            game.Add("Cricket");
            game.Add("Golf");
            game.Add("Gymnastics");
            game.Add("Tennis");
            game.Add("Hockey");
            return game;
        }
    }
}
```

Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager.

The following example for remote data binding using OData service.

CSHTML

```
<div id='o-data' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>
dataManger.Url("https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Order
s/").Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
    {
        Value = "CustomerID"
```

```

    }).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
</div>
</div>

```

ODATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web.Mvc;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```

Offline mode

To avoid post back for every action, set the autocomplete to load all data on initialization and make the actions process in client-side. To enable this behavior, use the **Offline** property of **DataManager**.

CSHTML

```

<div id='o-data' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("http://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders
/").Offline(true).Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
    {
        Value = "CustomerID"
    }).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
    </div>
</div>

```

OFFLINE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{

```

```

public class AutoCompleteController : Controller
{
    public ActionResult offline()
    {
        return View();
    }
}

```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Templates in ASP.NET MVC AutoComplete Control

The AutoComplete has been provided with several options to customize each list items, group title, header and footer elements. It uses the Essential JS 2 **Template engine** to compile and render the elements properly.

Item template

The content of each list item within the AutoComplete can be customized with the help of [itemTemplate](#) property.

CSHTML

```

@Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("O
DataV4Adaptor").CrossDomain(true)).ItemTemplate("<span><span
class='name'>${FirstName}</span><span class
='city'>${City}</span></span>").Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "FirstName"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
.city {
    right: 15px;
    position: absolute;
}
</style>

```

ITEMTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;

```

```
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public IActionResult Itemtemplate()
        {
            return View();
        }
    }
}
```

Group template

The group header title under which appropriate sub-items are categorized can also be customized with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).GroupTemplate("<strong>${City}</strong>").Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "FirstName",
    GroupBy = "City"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6).where(new ej.data.Predicate('City', 'equal', 'london').or('City', 'equal', 'seattle'))").Render()
```

GROUPTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public IActionResult grouptemplate()
        {
            return View();
        }
    }
}
```

Header template

The header element is shown statically at the top of the suggestion list items within the AutoComplete, and any custom element can be placed as a header element using [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("<span class='item'
><span class='name'>${FirstName}</span><span
class='city'>${City}</span></span>").HeaderTemplate("<span
class='head'><span class='name'>Name</span><span
class='city'>City</span></span>").Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "FirstName",
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
    .head, .item {
        display: table;
        width: 100%;
        margin: auto;
    }
    .head {
        height: 40px;
        font-size: 15px;
        font-weight: 600;
    }
    .name, .city {
        display: table-cell;
        vertical-align: middle;
        width: 50%;
    }
    .head .name {
        text-indent: 16px;
    }
    .head .city {
        text-indent: 10px;
    }
</style>
```

HEADERTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
```

```

        public ActionResult headertemplate()
        {
            return View();
        }
    }
}

```

Footer template

The AutoComplete has options to show a footer element at the bottom of the list items in the suggestion list. Here, you can place any custom element as a footer element using [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the AutoComplete.

CSHTML

```

@Html.EJS().AutoComplete("games").DataSource((IEnumerable<object>)ViewBag.data).Placeholder("Select a game").PopupHeight("270px").FooterTemplate("<span class='foot'> Total list items: " + 5 + "</span>").Render()
<style>
    .foot {
        text-indent: 1.2em;
        display: block;
        font-size: 15px;
        line-height: 40px;
        border-top: 1px solid #e0e0e0;
    }
</style>

```

FOOTERTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult footertemplate()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball", "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}

```

No records template

The AutoComplete is provided with support to custom design the suggestion list content when no data is found and no matches are found on search with the help of [noRecordsTemplate](#) property.

CSHTML

```
@Html.EJS().AutoComplete("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).No
RecordsTemplate("<span class='norecord'> NO DATA AVAILABLE</span>").Render()
```

NORECORDSTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public IActionResult norecords()
        {
            ViewBag.data = new string[] { };
            return View();
        }
    }
}
```

Action failure template

There is also an option to custom design the suggestion list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svcs/").Ad
aptor("ODataV4Adaptor").CrossDomain(true)).ActionFailureTemplate("<span
class='action-failure'> Data fetch get fails</span>").Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "FirstName"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName']).take(6)").Render()
```

Note: [View Sample in GitHub.](#)

See also

- [How to acheive filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)

Virtualization in AutoComplete Component

AutoComplete virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a AutoComplete activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

When the enableVirtualization property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The AutoComplete can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `text` column from complex data have been mapped to the `value` field.

CSHTML

```
@Html.EJS().AutoComplete("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>) ViewBag.data).En
ableVirtualization(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "Text"
}).Render()
```

VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult virtualscroll()
        {
            ViewBag.data = new Record().RecordModelList();
            return View();
        }
    }
}
```

```
}
```

Binding remote data

The AutoComplete supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

CSHTML

```
@Html.EJS().AutoComplete("records").Placeholder("Select a  
item").PopupHeight("200px").DataSource(obj  
=>obj.Url("https://ej2services.syncfusion.com/aspnet/development/api/orders"  
).CrossDomain(true).Adaptor("WebApiAdaptor")).EnableVirtualization(true).All  
owFiltering(true).Fields(new  
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value =  
"OrderID"}).Render()
```

VIRTUALSCROLL.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using WebApplication1.Models;  
namespace WebApplication1.Controllers  
{  
    public class AutoCompleteController : Controller  
    {  
        public ActionResult virtualscroll()  
        {  
            ViewBag.data = new Record().RecordModelList();  
            return View();  
        }  
    }  
}
```

Grouping

The AutoComplete component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the **groupBy** field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding virtualization, enhancing performance and responsiveness.

The following sample shows the example for Grouping with Virtualization.

CSHTML

```
@Html.EJS().AutoComplete("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).En
ableVirtualization(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { GroupBy = "Group",
Value = "Text" }).Render()
```

VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult virtualscroll()
        {
            ViewBag.data = new Record().RecordModelList();
            return View();
        }
    }
}
```

Grouping in ASP.NET MVC AutoComplete Control

The AutoComplete supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed as both inline and fixed headers. The fixed group header content is updated dynamically on scrolling the suggestion list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

CSHTML

```
@Html.EJS().AutoComplete("Vegetable").Placeholder("Select a
Vegetable").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.dat
a).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value =
"Vegetable", GroupBy = "Category" }).Render()
```

GROUPING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {

```

```

        ViewBag.data = new Vegetables().VegetablesList();
        return View();
    }
}
}

```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [groupTemplate](#) property for both inline and fixed headers as referred here:

[Group Template support to AutoComplete.](#)

Note: [View Sample in GitHub.](#)

Filtering in ASP.NET MVC AutoComplete Control

The AutoComplete has built-in support to filter data items. The filter operation starts as soon as you start typing characters in the control.

Change the filter type

Determines on which filter type, the control needs to be considered on search action. The available [filterType](#) and its supported data types are as follows.

Filter Type	Description	Supported Types
---	---	---
StartsWith	Checks whether a value begins with the specified value.	String
EndsWith	Checks whether a value ends with the specified value.	String
Contains	Checks whether a value contains the specified value.	String

The following examples shows the data filtering is done with **StartsWith** type.

CSHTML

```

@Html.EJS().AutoComplete("customers").Placeholder("Select a customer").PopupHeight("200px").FilterType(Syncfusion.EJ2.DropDowns.FilterType.StartsWith).DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "ContactName",
}).Query("new
ej.data.Query().from('Customers').select(['ContactName'])").Render()

```

FILTERTYPE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers

```

```
{
    public class AutoCompleteController : Controller
    {
        public ActionResult filtering()
        {
            return View();
        }
    }
}
```

Filter item count

You can specify the filter suggestion item count through [suggestionCount](#) property of AutoComplete.

The following example, to restrict the suggestion list item counts as 5.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a customer").PopupHeight("200px").SuggestionCount(3).DataSource(dataManger =>
dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "ContactName",
}).Query("new
ej.data.Query().from('Customers').select(['ContactName'])").Render()
```

FILTERCOUNT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult filtering()
        {
            return View();
        }
    }
}
```

Limit the minimum filter character

You can set the limit for the character count to filter the data on the AutoComplete. This can be done by setting the [minLength](#) property to AutoComplete.

In the following example, the remote request doesn't fetch the search data, until the search key contains three characters.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").MinLength(3).DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
    {
        Value = "ContactName",
    }).Query("new
ej.data.Query().from('Customers').select(['ContactName'])").Render()
```

FILTERLIMIT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult filtering()
        {
            return View();
        }
    }
}
```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by setting the [ignoreCase](#) property of AutoComplete.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a
customer").PopupHeight("200px").FilterType(Syncfusion.EJ2.DropDowns.FilterTy
pe.StartsWith).IgnoreCase(false).DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
    {
        Value = "ContactName",
    }).Query("new
ej.data.Query().from('Customers').select(['ContactName'])").Render()
```

CASESENSITIVE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Diacritics filtering

An AutoComplete supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

CSHTML

```
@Html.EJS().AutoComplete("Diacritics").DataSource((string[])ViewBag.data).Placeholder("e.g: aero").IgnoreAccent(true).Render()
```

Note: [View Sample in GitHub.](#)

See also

- [How to achieve autofill while filtering](#)
- [How to group the data using header](#)
- [How to highlight the search data](#)

Localization in ASP.NET MVC AutoComplete Control

The Localization library allows you to localize static text content of the `noRecordsTemplate` and `actionFailureTemplate` properties according to the culture currently assigned to the AutoComplete.

| Locale key | en-US (default)

|-----|-----

| noRecordsTemplate | No Records Found

| actionFailureTemplate | The Request Failed

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the AutoComplete and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

CSHTML

```
@Html.EJS().AutoComplete("customers").Placeholder("Select a customer").PopupHeight("200px").Locale("fr-BE").DataSource(dataManger =>
    dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
```

```
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings
{
    Value = "ContactName"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(0)").Render()
<script>
    ej.base.L10n.load({
        'fr-BE': {
            'dropdowns': {
                'noRecordsTemplate': "Aucun enregistrement trouvé",
                'actionFailureTemplate': "Modèle d'échec d'action"
            }
        }
    });
</script>
```

LOCALIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult localization()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

See also

- [Accessibility](#)
- [How to bind the data to the autocomplete](#)

CSS structures in AutoComplete Control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
```

```
.e-ddl.e-input-group.e-control-wrapper .e-input {
```



```
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}
`
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}
`
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-
input-focus::after {
background: #c000ff;
}
`
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-
success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}
`
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css
.e-input-group.e-control-wrapper .e-input[disabled] {
-webkit-text-fill-color: #0d9133;
}
`
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
background-color: #2319b8;
}

.e-ddl.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-float-
input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
`
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css
.e-ddl.e-input-group input.e-input::placeholder {
color: red;
}
`
```

Customizing the placeholder to add mandatory indicator(*)

Use the following CSS to add the mandatory indicator * to the float label element.

```
`css
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after {
content: "*";
color: red;
}
`
```

`

Customizing the text selection color

Use the following CSS to customize the selection color of text and background.

```
`css
.e-ddl.e-input-group input.e-input::selection {
color: red;
background: yellow;
}
```

`

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
```

`

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
```

`

Note: [View Sample in GitHub.](#)

Accessibility in ASP.NET MVC AutoComplete Control

The AutoComplete control has been designed, keeping in mind the **WAI-ARIA** specifications, and applies the **WAI-ARIA** roles, states, and properties along with **keyboard support**. This control is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The AutoComplete component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the AutoComplete component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The AutoComplete control uses the **combobox** role and each list item has an **option** role. The following ARIA Attributes denote the AutoComplete state.

| Property | Functionalities |**| --- | --- |****| aria-haspopup |** Indicates whether the AutoComplete input element has a suggestion list or not. **|****| aria-expanded |** Indicates whether the suggestion list has expanded or not. **|****| aria-selected |** Indicates the selected option from the list. **|****| aria-readonly |** Indicates the readonly state of the AutoComplete element. **|****| aria-disabled |** Indicates whether the AutoComplete control is in a disabled state or not. **|****| aria-activedescendent |** This attribute holds the ID of the active list item to focus its descendant child element. **|****| aria-owns |** This attribute contains the ID of the suggestion list to indicate popup as a child element. **|****| aria-autocomplete |** This attribute contains the 'both' to a list of options shows and the currently selected suggestion also shows inline. **|****Keyboard interaction**

You can use the following key shortcuts to access the AutoComplete without interruptions.

| Keyboard shortcuts | Actions |**| --- | --- |****| Arrow Down |** In popup hidden state, opens the suggestion list. In popup open state, selects the first item when no item is selected, else selects the item next to the currently selected item. **|****| Arrow Up |** In popup hidden state, opens the suggestion list. In popup open state, selects the last item when no item is selected, else selects the item previous to the currently selected one. **|****| Page Down |** Scrolls down to the next page and selects the first item when popup list opens. **|****| Page Up |** Scrolls up to previous page and select the first item when popup list open. **|****| Enter |** Selects the focused item and set to AutoComplete control. **|****| Tab |** Focuses on the next tab indexed element when the popup is closed. Otherwise, closes the popup list and remains the focus in control suppose if it is in an open state. **|****| Shift + tab |** Focuses the previous tab indexed element when the popup is closed. Otherwise, closes the popup list and remains the focus in control suppose if it is in an open state. **|****| Alt + Down |** Opens the popup list. **|****| Alt + Up |** In popup hidden state, opens the popup list. In popup open state, closes the popup list. **|****| Esc(Escape) |** Closes the popup list when it is in an open state then remove the selection. **|****| Home |** Cursor moves to before of first character in input. **|****| End |** Cursor moves to next of last character in input. **|**

Note: In the below sample, focus the AutoComplete control using alt+t keys.

CSHTML

```
@Html.EJS().AutoComplete("games").Placeholder("Select a game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Render()
<script>
    document.onkeyup = function (e) {
        if (e.altKey && e.keyCode === 84 /* t */) {
            var dropObj = document.getElementById("games"); //to get dropdown list object
            dropObj.ej2_instances[0].focusIn(); // call the focusIn method to focus the element.
        }
    };
</script>
```

ACCESSIBILITY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult accessibility()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball", "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Ensuring accessibility

The AutoComplete component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the AutoComplete component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the AutoComplete component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

How To

Autofill supported with AutoComplete

The AutoComplete supports the autofill behavior with the help of [autofill](#) property. Whenever you change the input value, the AutoComplete will autocomplete your data by matching the typed character. Suppose, if no matches were found, then AutoComplete doesn't suggest any item.

In the below sample, showcase that how to work **autofill** with AutoComplete.

CSHTML

```
@Html.EJS().AutoComplete("games").Autofill(true).Placeholder("Select a game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "Name" }).Render()
```

AUTOFILL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult autofill()
        {
            ViewBag.data = new Countries().CountriesList();
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Show the list items with icons

You can render **icons** to the list items by mapping the [iconCss](#) field. This **iconCss** field creates a span in the list item with mapped class name to allow styling as per your need.

CSHTML

```
@Html.EJS().AutoComplete("icons").Placeholder("Select a social media").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.icondata).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "SocialMediaName", IconCss = "Class" }).Render()
<style>
@@font-face {
    font-family: 'Socialicons';
    src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMvltCfsAAAEoAAAAVmNtYXCNKKeOAAABrAAAAEhnbHlml19XagAAAgwAABhQaGVhZA8dCeEAAADQAAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAAAAYAAAAAAsbG9jYR3AIwWAAA0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGX5D00AABY0AAAAkwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAACwABAAAAAQAA+iTiP18
```

PPPUACwQAAAAAANYFYngAAAAA1gVieAAAAAAD9AP0AAAACAACAAAAAEEAAAAALafQACwAAAAA
AAgAAAAoACgAAP8AAAAAQAQAAZABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaaWAAABQAawABAAAAFAA
EADQAAAAEAAQAAQAApwn//wAApWd//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAAiQ
CzgOMBU4F/gZYB9QICaO+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fHQEfDhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLWYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kIw8FAYkFEgQDAyQDAQECAyIBAQMSEgkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgWKCCQEBCICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwCfBQUEAwICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRUVFBU
VDw4ODg0NDawLCwoKCQkICACGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwwNDQ0ODg4PDw8QEBA
QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwwNDQ4ODxAREhISEhMTEExMUExQUFRQ
VGXsaGgcIBwfxNgEBAQ8KCgoIBwcGBQUdAwIBAQECaWMDbQUFBGcHCAGICgkKCwsMDAwNDg0ODw8
PEBAQEHISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCACh/aaQEB0cGhgWFBIRDgw
LCaCEAwICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQEA8ODhAQDQ0LCQg
HBgQCAgICBAQEBgYGCAGJCQoLCwwMDQ4ODg8PEBAREBESEhISExITGBcZGRgZGBgXAU4CAGMEXAk
FBAQFBCQCAwMGHcKfQkMIwIBAwYAwIBKQECBSgFBgULCgkHBgMBAQEDAwICGwMDg8PEhITFBU
WGBgSEyYJCAgIBwcHDBEGAEBAAgEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhE
SEREREBEPEA8QDhMEBASFNgEBAQEJCAcGBQMCAQEDAwUGCAgHCAgJCQoKCwsMDA0NDQ4ODg8ODxA
PEA8QEBAREBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICACGBgUFAwMDAQEBAQEC6RISExISExISEhM
SEhESERERERAQEA8QDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVNcFBAQVEBEREhESEhI
TEhMTFBMUEhEREBEQEBApDw8PDg4ODQ0MDAwLCwoKCgkICACHBWUGBQQDAgIBAQEBAgIDBAQFBQc
GCAGICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUfHcYGRScHR8fIiIjFBMTEhMSEhIREhEREBEQEAM
EAwt1YQINCAyEAgIEBAUGBgICQoKDAwNDg8PERUVFhcWGBcYGBkZGRkaGxoTEhISEhEREBAQDw8
ODg4MDQwLCgoKCQgHBWYFBQMDAwEBAGMFBQgIAAAAAEAAAAAAzoD9ACWAAATDwYVERUfHTsBPw4
9AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwwMDA0NDQ0ODg4PDw8
QEBDQCGsKCQkJCAGHBWUEBAICAgIEBAUHBwgICQkJCgsK0QoKCgoJCAgIBwcFBAMDAQEBKQoJCgg
JCAgHBWUFBAQCAQEBAQIEBAUFBwgHCAkJCQkK/tcCAGMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgs
KDAwMDf4LEhMTehESEREQEBApDw4PDQ4MDA0KCQgIBgYEBAMCAgEBAwQFBwcJCQoKCwsMDA0NDAs
MCwoKCQkHBWUEAwEBAQEDBAUGCAgJCgoLCwwMDVgCAwMFBgcICQkJCgsLCwwLDAsKCgoJCAgHBgU
EAgIBtQ0MDAsLCgoJCQcHBQDAQEBAQMEBQYIAAABAAAAAP0A90AqAAAT8DMx8MHQEPPDSsBLxo
PCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBWUFbQDQAwICAgEDBQoPEXyWFAsLCgQFBAUFBAU
FBQUJCQkJCYQEBOUGBwcICAKKCgsLDQwODQ8RERQfI5IvNRMGBWYGBWYGBgYGDAsMWAcICAgICQk
JCQkKCgoLCgsJEXITFBQVFRYWFIMkJTEWFRQREQ8NDAsJBWYFAwEBAQEDBAUHBwkJCwwNDhAQEGs
YGBYWFbQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBQUHBgcICBMSDxEdIiUoIXsOCgcCAQIEBAYICBU
bHyU37xQTERAPDQwKCQgGBQMCAQEDBgLDhofkUInCwIBAgQEBwcJCwscISf+pBYUEXIQDg4LCwk
IBgUDAgEDBACJDA0REhQXJysxRiEhIB4eHRwbGhkZfxcVFRoXfHqTERAODQwKCAcGBAMBAQMFBwk
MDQ8REXUXGhsdAAUAAAAA/ED9ABCAKoA6wESAYQAAEdAQ8NKwEvDjU/EB8OJR0BHW8hPw8TLwM
hHwUVDxEvEzU/CSchDwMFFR8PPw8vDw8OAR8HFQ8JIY8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM
9Ai8BIw8EFQ8DIY8CNw8KFxUfASUzPwgZHWkhPwI1LxAlDwICkAMDBQcHCQkKDAwMDQ4ODwwMCws
LCgoJDwsJCAyFAGCAwQGBgcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBgkJDAwODwg
RERITEwJpExMSEhEQDw4MDAUJBWYEAgEBAgEF/uYOCwkGBAICBAYHCQsMDg8QERETExQTFRQVFBQ
UFBMTEhAPDg0LCQgGBAMCAgEBAwMEBQYHCAKc/uoFAGEBASwBAWUGCQoLDQ0PERESEhQUFBQTEhE
QEA4MDAKJBgUDAQEDBQcICgsNDg8QERISFBQUFBMSERAQDgWMCgghBAMCPAYGBgQEAWEBABQEBAM
EBAQFbmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFeAKDAwECBAQEBAQDAWI
CAWUWAwIBAQQDdwLCQgFAwEBAQEEATEEBYUFRYWFxcYFxcXFxYVFBgFBQYBJwYCAgICBAYHCQo
LDA4ODxAIERIR/d8FAQIB9wcHDg0NDawLCgkIBwYFBAICAgQEBQYGDQwODg8REBINDQwMCwsKCgk
IBwcGBAQCAgEYFbgICgoLDQ0NDg9929sUEXISERAPDgWMBQkHBgQCAQMFbgkJDAwODwgQERIE
TEWHBGBMBARYXFxcXFfWfUUFBMREQ8ODAsJCAYEAWIBAgQFBWkKDA0PDxEREXQPEA8PDw8PDw8
ODw4ODg4OAQEBAQKPCgoUEhIREBANDQsKCAcFAwEBawUHCAoLDQ4PEBESExQUFBMTEhEQDw4NCwo
IBWUDAQEDBQcICgsNDg8QEHITEWGSAQICBAUFBgdsBQQFBAQDAgIBAQECAGQFBQYHawCHBgUDAgE
BR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDAwmG0ggEAQICAgTKAQ00EBASEhQ
VEiRdAgIBAQITDg0JCAyDAQQFBwoMDhQCAQEBAQNuJBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAM
AAAAA/QD3QADAFcAlwAANzMRIwUVIzclIXeZET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQH
W1gIDBQgKCwCHBwgJCQoKCw4NDAsKCAgHBWUEBAICAQHWAQICAgQDBQUFBgYHBwcJCAkJCgoKCws
LDBgZGhQUEREPDg0MCwoJCQ79xAEBawMFBgYHCAKKCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAY
GCAGJCgoLDQwODQ0MDA0KCQkHBWYEBAMBIgKFWwICW/17AXcUDA0ODgwGBQUEBAMCAQEBAgMFBQc

ICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQwMCwoKCQkICAcGBgUFBAMGAwEBAGMEBgYHCAgICQkSARI
MCwsKCgkICAgGBQUEAwEBAQEDBAUFBgICAKKCsLDAsLCwsJCggIBwYGBAQDAQEBAQMEBAYGBwg
ICgkLCwsAAAABAAAAAPuA/QARgAAExEVHwYhESM1MzU/DzMVIw8GFTMVIxEhPwYRLwYhDwYSAGQ
FBwgKCgHPb24BAwMGBggJCgsMDQ0ODwgPlUcLCwkIBgQDe3sBBQoKCACFBAICBAUHCAoK/IUKCgk
HBwQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBwUEAgGFAwQHCAkKDDOF/mUDBAYICQoLA4I
LCgkIBgQDAQQFBwgKCwAAAAAGAAAAAP0A/QAOABEAIAABBQEQAuWAAAAPCR0Bhw07AT8NPQEvCCM
PASUVMxUjFSM1IzUzNSUPBRUfDTsBPww1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIH1y8NPQE
/DwEVHw8hPw8RITChLw8hDw4BCgMTCwsFBAQEAgICAwQGBgcICgoLDAwODg8NDQwLCgkICAYGBQQ
DAwEBAQIDBAGMDiYRNw0B9nR0TXNz/kAFawMDAQIBAgMDBAQGBgcICQkKDasIBwCHBwYFBQYFAwM
BAQECAwMEBQYGBwgJCQoLDACIBwCHBwX+MTAQDggIAwICAQEBAQEDAwMICgsMDAsGAgEBAQECAwY
iGQoFCQcDAgIBAwQFCAGLDA0PERITFRYYFRISEA8NDAsKCACGBAMCAQEBAwUHCQsOERMUFB0xCAC
DAwEBAQIFGQ4ODQ0LCgoICAcFBQCCAgMDBgcICgwICBESEhESEBD+pwEDBQYJCgsNDg8IEBISEXQ
CahQTExiREA8ODQsGCQcGBAL8GAED5gIDBgCICgsNDg4QCBIREXP9lhMTExEREA4ODQsKCACGAwF
KAQkHCAYGBggICQkKCgkICAgHBgYFBQMDAgIBAgMDBAUFBgYHBwCICQgHBwYGBgYLCwwCBQPYck9
yck5zZwYGBwCHDxELCgWLCwsKCgkJBwUFAwECAwMDBAUHBwCIBw0QCwwLDAsMCgoKCACGBAMBAgI
DAwQFLRkQDwwPCAgJCgoLCQkICAgNDAsKCQwJBQYGBQYEBACbFQsGDA4HCAgJCQ4NDQwNCwwKCgg
IBgYDAwEBAgMDBQYGBwgJCAoJCgsKCwUMDAwMDAsKCQYFBQUKDAYHCAgJBw0BAGQEBQcHCAkJCgo
KCwsLDQ4NDQ0MDAsGBgkIBQQCAQH+EAoKExMSERAQDQ0LBgkHBgQCAQMFbgkKCw0NEAgQEhITFAJ
HKxQSEhIQDw8NDAsJBQcFBAIBAwQHCAkLDA0PDxASEhIAAAAAAGAAAAAD7gP0AEAAhAAAAARUzFSM
RHws/BxUPAy8OESM1Pw81ER8OMyEzPw4RLw4jISMPDQIbysoDBgUICgYHCAgJCgsLDQ4PEBESE0Q
tICIiEREQDw8ODQwKCgCHBANuGBkVDw4ODgYFBgUEBAMCAv5fAQECAwQEBQUGBwCHCAgJCAM0CAk
ICACHBwYFBQQEAWIBAQEBAgMEBAUFBgCHBwgICQj8zAgJCAgHBwCGBQUEBAMCAQON0H/+9BIMCAk
HBAMDAgEBAQEBAgMDBQYHeA4GAwEBAgIDBAUFBgJCwsNDxABVGwKDXANDxEUCwwMDQ0ODxAQEvz
CCQgICACHBwYGBAUDAwICAgIDAUEBgYHBwCICAgJAz4JCAgIBwCHBgYEBQMDAgICAgMDBQQGBgc
HBwgICAAAAGAAAAAD7APzAPgBqAAAAAR8LFQ8MIy8QKwEPDh8bHQEPfi8WPQE/DTMfEjM/Di8ePQE
/Fh8CBR8HDwMfhjSBPwIfBzM/HTUvBz8CPQEvHiMPai8HIw8dAnALFhMSDw4LCQgFBAIBAgIDAww
FBgUGBgGCawLCQgHChQLCwsHBwkJCgsNDQwMCwsJCggIBwYFBAMDAQEBAgMEBQcHCRMTdxojFhQ
TEA8OCwUFAwQCAwEBAgIEBQUHCAgKCgWMDg4PEBEREHMTFBUZGBYWFRMSEgSLCwoJCQgIBwYFBQM
CAgECAgMDBAUFBQYGBgYHCAsLCgkIBwCMBwCHBwKDAcPERMZDQ0MDAsKCQgHBgUEAwEBAQICAgM
EBAsMDQ8bTSIfGxkMCwsKCQgIBwYFBQMCAGICBAQGBggICQoLDA0NDw8PERERExIUHxwb/bsBAgM
EBQcHCQUDAQEBAQMFbQYICakLCwwNDg8QEBESEhMUFBWFRcWGBcYGBYWFRUPDXAQEBEREQ4ODg0
NDQ0MDAwMCwoLCgkJCQgHBwCGBgQFAwMDAgEBAQIDBAUGBgQEAgIDBAUHBwkJCgWMDQ4PDxERERM
TFBQVFRYWFxcYGBgUFRQTEBESEhITFBMODg4NDQ0NDA0LDAsKCwoJCQkIBwCHBgYEBQMDAwIBAzC
ECAoLDAwNDQ4NDg0NBgYGBQYKBQQDAwICAQECAUHDSEODQoEBAMCAgIBAQICAwMEBQUFBQYGBgY
GCACHBgYFBQUIBx0GDAgJCgsNDg8JCAkKCsLCwwPDg0ODQwMDAsLCgoICAgHBgUEBAMCAQEBAgI
EBQYICAYIBwkJCQoKCwsLCgsLCgoHBgYGBQUFBQQEAWMCAQEBAgUGCAkLGg0LCgkICAYDBAMCAQI
DBAQFBgYGBwCIBwkIDQcFBgUEBQgIBgYHEgkJCgoHBgcICAKJCgoLDAwMDg0NDQ0MDAsLCgoKCQg
IBwYGBQQEAWMBAQEBAwRbEhMSEREREBAXFgYGBgYFxcWfHUVFBQTExEREQ8PDg0MDAoKCAChBQq
DAgICAwCGBgUDAwEBAQIDAwmFBAYGBwCHCAkJCQoLCgsMDAwMDQ0NDQ4ODhAQEA8PDw4OGB0ZGhg
YFxfGfXUWFRQUEXISERAQDw4NDAsLCQkHBgUFAwEBAgIDCggHBgUDAgEBAgMDAwUEBgYGCACICQk
JCgoLCwwLDQwNDQ0NDg4AAAAACwAAAAAD8wOYABEAMwBbAKYAYwDTARcBOQFjAZgBoQAAAQ8DMzc
vBisBDwEnDwIdAh8FOWE/BjUvBisBDwEnFwcfBDM/Bic1MxUnNw8GIy8HNyclHwsVIxUfBjsBPwY
1MxcVDw0vCzU/CycVPwMfCR0CDwgjLwQPAREjFSMVIzUjNTcPCxUfDyE/DzUvDiMhIw8BJR8DFQ8
GKwEvBjU/Bx8DFR8KPwUHMzUjFQ8GKwEvBjUjDwcdAR8LowE/CTUvDg8DFTM1NyMHJyMDIgdDagJ
CAgECAwQFBQYGCgUG2QQDAgIDBAUFBgYGBQYEBQICAQECAgUEBgUGBgUF6wEBawUCAwMEBwGEAgE
BAQFFOQEDBAYMDhAQDwgGBgYFagIEAQECHQoLCgkJCACFBAMCAXcBAQMDBQQFBhAGBQQEAgIBMwM
BAQMCBAQMBwCICAgIDxAPDg8HBgYEAwMBAQECAgQGBggJCQkJC+UQDg4NDQUGBwCFBAQCAgIDAww
GBgcICQoLBQwNFAQ50VJFRyAPDQ0LCwkJBwMFAwIBAgQGBwkJCwsNDQ8PDxARAqAQEQ8PDw0NCws
KCAcDGBQCAQIEBgCICgsLDQ0PDxAQEP1gERAPAYoEAgIBAQICBAUGBQYGBgUFBAlCAQECAwQFBQU
GBgUDgdgEEAgQEBQQGBwgIBwCGBgoKAUw7AQEDAwQEBQUFBQQEAWIBAUC+CggGBgMCAgICBAMECgY
HCQsLCwwMCwoSBwCHCAUEAgEBAgIEAwQHBwgJCgsMDg8NDMPKv1AuLVEBbAMEBB8bBQMEAwICAQE
CCAMDAwOAAwMDAgICAQECAgIDAwwAAwMDAwICAQECF4kgBQUCAQECAQEQAQIDCJrYARsEBAMHBQQ
CAQICAwQEBRoPmw0BAQIDAwwQFBgYMDBgvmgYEAwIDAQEBAQMCAwQEGREGBgUFBQQECCQEAWICAQE
BAwYHBQYGBwgJCgpDDwwLCggHBgYDAwMBQFQBHQBAQICAwUFBQYHBwhwCgkJCAgGBQQDAQEBAU
LEgEBIib+/yVJBQUGBwCICQkFCgsK9gsLCgoJCQgHBwYFBQMDAQEBAQMDBQUGBwCICQkFCgoL9wo
LCgoJCQgHBwYFBQMDAgID9wQEBAV3BAUDBAMCAQECAwQEBAR3BQQEBAMCAQEBAQJ3GAwRBAUEAwM
DAQEBAQEBAwCkEuCvAwMDAgMBAQEBAwIDAwwOvAgUHCAoKDA00QxgOBwCgCgQEAgMCAQICBQQFBws

```

KDxZTCgkHBgYGBQYFBQMDAgEBAQIEPayslG9vAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAAAAABAA
AAQABAAAAAAAAACAACADAABAAAAAAAAADAASAEwABAAAAAAAAEAAASAHgABAAAAAAAAFAAAsAKQABAAAAAA
GAASANAABAAAAAAAKACwAPwABAAAAAAALABIAawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAE
ECQACAA4AlQADAAEECQADABYAowADAAEECQAEABYAuQADAAEECQAFABYAzWADAAEECQAGABYA5QA
DAAEECQAKAFgA+wADAAEECQALACQBUBYt2NpYWxpY29uc1JlZ3VsYXJt2NpYWxpY29uc1NvY21
hbGljb25zVmVyc2lvbiAxLjBTb2NpYWxpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAUwBvAGMAaQBhAGwAaQBjAG8AbgB
zAFIAZQBnAHUAbABhAHIAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAFMABwBjAGkAYQBsAGkAYwBvAG4
AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAEYAbwBuAHQAIAB
nAGUAbgBlAHIAIYQB0AGUAZAAGAHUAacwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIABNAGU
AdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQA
AAAAACAAAAAAAOAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAsBAgEDAQQBBQEGAQCBCAEJAQoBCwE
MAAh3aGF0c2FwcAd0d2l0dGVyBXZpbWVvCWLuc3RhZ3JhbQhsaW5rZWRpbgghmYWNlYm9vawtnb29
nbGUTCgxlwZ0dWlibHIIc2t5cGUtMDEIeW9ldHViZTEAAAA=) format('true');
    font-weight: normal;
    font-style: normal;
}
.e-list-icon {
    font-family: 'Socialicons' !important;
    color: rgba(0, 0, 0, .57);
}
.twitter:before {
    content: "\a701";
}
.vimeo:before {
    content: "\a702";
}
.youtube:before {
    content: "\a709";
}
.whatsapp:before {
    content: "\a700";
}
.skype:before {
    content: "\a708";
}
.instagram:before {
    content: "\a703";
}
.google-plus:before {
    content: "\a706";
}
.facebook:before {
    content: "\a705";
}
.tumblr:before {
    content: "\a707";
}
.linkedin:before {
    content: "\a704";
}
}
</style>

```

ICONS.CS

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult icons()
        {
            ViewBag.icondata = new SocialMedia().SocialMediaList();
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Custom highlight search

The AutoComplete has built-in support to highlight the searched characters on suggested list items when the [highlight](#) property is enabled.

The below sample customizes the matched character in suggestion list by `e-highlight` class.

CSHTML

```
@Html.EJS().AutoComplete("games").Highlight(true).Placeholder("Select a game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "Name" }).Render()
```

HIGHLIGHT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult highlight()
        {
            ViewBag.data = new Countries().CountriesList();
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Filter using both text and value field

The AutoComplete data can be filtered based on both text and value fields using predicate of dataManager through filtering event. The filtered data can be again updated through updateData method.

CSHTML

```
@using Newtonsoft.Json;
@Html.EJS().AutoComplete("project").DataSource((IEnumerable<object>)ViewBag.data).Fields(new Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Text = "Name", Value = "Code" }).ItemTemplate("<span class='item'><span class='name'>${Name}</span>-<span class='code'>${Code}</span></span>").Filtering("filtering").Render()
<script>
    function filtering(e) {
        var predicate = new ej.data.Predicate('Name', 'contains', e.text, true);
        predicate = predicate.or('Code', 'contains', e.text);
        var query = new ej.data.Query();
        query = (e.text !== '' && e.value !== '') ? query.where(predicate) : query;
        e.updateData(@Html.Raw(JsonConvert.SerializeObject(ViewBag.data)), query);
    }
</script>
```

FILTERING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult filtering()
        {
            ViewBag.data = new Countries().CountriesList();
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

AutoCompleteFor

The AutoCompleteFor control can be rendered by passing values and data from the model. The selected values can be retrieved during form submit using the post method.

CSHTML

```
@using (Html.BeginForm())
```

```
{
    @Html.EJS().AutoCompleteFor(model => model.value).Placeholder("Select a
Country").DataSource((IEnumerable<object>)ViewBag.data).Fields(new
Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "Name"
}).Render()
    <div id="submitButton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}
```

FOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        Countries model = new Countries();
        public ActionResult Index()
        {
            ViewBag.data = new Countries().CountriesList();
            model.Value = "Cameroon";
            return View(model);
        }
        [HttpPost]
        public ActionResult Index(Countries model)
        {
            ViewBag.data = new Countries().CountriesList();
            model.Value = model.Value;
            return View(model);
        }
    }
}
```

Data Annotation

Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end users.

Data Annotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET Framework will automatically enforce these validation rules and display validation messages in the view.

Using **value** property gets or sets the value of the selected item in the control.

CSHTML

```
@using (Html.BeginForm())
{
    @Html.EJS().AutoCompleteFor(model =>
model.EnquiringAboutSelect).Placeholder("Select a
Country").DataSource(Model?.EnquiringAboutSelectListItems).Fields(new
```

```

Syncfusion.EJ2.DropDowns.AutoCompleteFieldSettings { Value = "Text"
}).Placeholder("Please Select Enquiring About").Render()
    <div>
        @Html.ValidationMessageFor(model => model.EnquiringAboutSelect)
    </div>
    <div id="submitButton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}

```

FOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {
            DataModel model = new DataModel();
            model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
            model.EnquiringAboutSelect = "Beacon";
            return View(model);
        }
        [HttpPost]
        public ActionResult Index(DataModel model)
        {
            model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
            model.EnquiringAboutSelect = model.EnquiringAboutSelect;
            return View(model);
        }
    }
    public class DataModel
    {
        [Required(ErrorMessage = "The value is Required")]
        public string EnquiringAboutSelect { get; set; }
        public List<ListItems> EnquiringAboutSelectListItems { get; set; }
    }
    public class ListItems
    {
        public string Text { get; set; }
        public string Value { get; set; }
        public List<ListItems> getListItems()
        {
            List<ListItems> items = new List<ListItems>();
            items.Add(new ListItems() { Text = "Aberdeen", Value = "103" });
            items.Add(new ListItems() { Text = "Alexandria", Value = "102"
});
            items.Add(new ListItems() { Text = "Albany", Value = "101" });

```

```

        items.Add(new ListItems() { Text = "Beacon ", Value = "104" });
        items.Add(new ListItems() { Text = "Brisbane ", Value = "105"
    });
    return items;
}
}
}

```

Migration from Essential JS 1

This article describes the API migration process of AutoComplete component from Essential JS 1 to Essential JS 2.

Note: MultiSelect concept is not present in EJ2-AutoComplete. If you want to use multiselection support in autocomplete, we suggest you to use MultiSelect component.

DataBinding

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property:** *Datasource*

`
@Html.EJ().Autocomplete("autocomplete").Datasource((IEnumerable<CarsList>)ViewBag.datasource)|` **Property:**

DataSource`
@Html.EJS().AutoComplete("games").DataSource((IEnumerable<object>)ViewBag.localdata).Render()|`

| **Fields for mapping** | **Property:**

AutocompleteFields`
@Html.EJ().Autocomplete("autocomplete").AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))|` **Property:** *e-autocomplete-*

fields`
@Html.EJS().AutoComplete("games").Fields(new AutoCompleteFieldSettings { Text = "Game" }).Render() |`

| **Query** | **Property:**

Query`
@Html.EJ().Autocomplete("selectCar").Query("ej.Query().from('Customers').take(10)")|` **Property:**

Query`
@Html.EJS().AutoComplete("customers").Query((string)ViewBag.query).Render()|`

| **Begin event** | **Event:**

ActionBegin`
@Html.EJ().Autocomplete("selectCar").ActionBegin("onBegin")|` **Event:**

ActionBegin`
 @Html.EJS().AutoComplete("customers").ActionBegin("onBegin").Render()|`

| **Complete event** | **Event:**

ActionComplete`
@Html.EJ().Autocomplete("selectCar").ActionComplete("onComplete")|` **Event:** *ActionComplete* `
`

`@Html.EJS().Autocomplete("selectCar").ActionComplete("onComplete").Render()|`

| **Failure event** | **Event:**

ActionFailure`
@Html.EJ().Autocomplete("selectCar").ActionFailure("onFailure")|` **Event:**

ActionFailure `
 @Html.EJS().Autocomplete("selectCar").ActionFailure("onFailure").Render()|`

| **Success event** | **Event:** *ActionSuccess*

`
@Html.EJ().Autocomplete("selectCar").ActionSuccess("onSuccess")|` **NotApplicable** |

Filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Case sensitivity | Property:

CaseSensitiveSearch
@Html.EJ().Autocomplete("selectCar").CaseSensitiveSearch(true) | **Property:** *ignoreCase*
@Html.EJS().Autocomplete("selectCar").IgnoreCase(true).Render() |

| Accent effective search | Not applicable | Property: *ignoreAccent*

@Html.EJS().Autocomplete("selectCar").IgnoreAccent(true).Render() |

| Filtering Type | Property:

FilterType
@Html.EJ().Autocomplete("selectCar").FilterType(SortOrder.None) | **Property:**

FilterType
@Html.EJS().Autocomplete("selectCar").FilterType(FilterType.Contains).Render() |

| Autofill | Property:

EnableAutoFill
@Html.EJ().Autocomplete("selectCar").EnableAutoFill(true) | **Property::**

AutoFill
@Html.EJS().Autocomplete("selectCar").AutoFill(true).Render() |

| Highlight the search word | Property: *HighlightSearch*

@Html.EJ().Autocomplete("selectCar").HighlightSearch(true) | **Property:** *Highlight*

@Html.EJS().Autocomplete("selectCar").Highlight(true).Render() |

| No of items to be shown | Property:

ItemCount
@Html.EJ().Autocomplete("selectCar").ItemCount(3) | **Property:**

SuggestionCount
@Html.EJS().Autocomplete("selectCar").SuggestionCount(3).Render() |

| Minimum characters to enter | Property: *MinCharacter*

@Html.EJ().Autocomplete("selectCar").MinCharacter(3) | **Property:** *MinLength*

@Html.EJS().Autocomplete("selectCar").MinLength(3).Render() |

| Search | Method: *search*

@Html.EJ().Autocomplete("selectCar")

\$("#selectCar").ejAutocomplete("search"); | **Not applicable** |

Placeholder

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Watermark text | Property: *WatermarkText*

@Html.EJ().Autocomplete("selectCar").WatermarkText("Select") | **Property:** *Placeholder*

@Html.EJS().Autocomplete("selectCar").Placeholder("select").Render() |

| Floating of waterMarkText | Not applicable | Property: *FloatLabelType*

@Html.EJS().Autocomplete("selectCar").FloatLabelType("Auto").Render() |

Popup

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **No records text** | **Property:** *EmptyResultText*

@Html.EJ().Autocomplete("selectCar").EmptyResultText("no records") | **Property:** *NoRecordsTemplate*

@Html.EJS().Autocomplete("selectCar").NoRecordsTemplate("@Html.Raw(" NO DATA AVAILABLE")").Render() |

| **No records showing** | **Property:** **show-empty-result-text*

@Html.EJ().Autocomplete("selectCar").ShowEmptyResultText("no records") | **Not applicable** |

| **Popupbutton** | **Property:** *ShowPopupButton*

@Html.EJ().Autocomplete("selectCar").ShowPopupButton(true) | **Property:** *ShowPopupButton*

@Html.EJS().Autocomplete("selectCar").ShowPopupButton(true).Render() |

| **Clear button** | **Property:** *ShowResetIcon*

@Html.EJ().Autocomplete("selectCar").ShowResetIcon(true) | **Property:** *ShowClearButton*

@Html.EJS().Autocomplete("selectCar").ShowClearButton(true).Render() |

| **Animation** | **Property:** *animate-type*

@Html.EJ().Autocomplete("selectCar").AnimateType("None") | **Not Applicable** |

| **Focusing the list item** | **Property:** *AutoFocus*

@Html.EJ().Autocomplete("selectCar").AutoFocus("true") | **Not applicable** |

| **Delaying the popup open time** | **Property:** *delay-suggestion-timeout*

@Html.EJ().Autocomplete("selectCar").DelaySuggestionTimeout(500) | **Not applicable** |

| **Popup text when there is no popup items** | **Property:** *empty-result-text*

@Html.EJ().Autocomplete("selectCar").EmptyResultText("no records")
| <https://ej2.syncfusion.com/aspnetmvc/AutoComplete/Template#/material> |

| **Enable/disable the duplicate option** | **Property:** *enable-distinct*

@Html.EJ().Autocomplete("selectCar").EnableDistinct(true) | **Not applicable** |

| **Popup height** | **Property:** *PopupHeight*

@Html.EJ().Autocomplete("selectCar").PopupHeight("300px") | **Property:** *PopupHeight*

@Html.EJ().Autocomplete("selectCar").PopupHeight("300px").Render() |

| **Popup Width** | **Property:** *PopupWidth*

@Html.EJ().Autocomplete("selectCar").PopupWidth("300px") | **Property:** *PopupWidth*

@Html.EJ().Autocomplete("selectCar").PopupWidth("300px").Render() |

| **Open popup** | **Method:** *open*

@Html.EJ().Autocomplete("autocomplete")

\$("#autocomplete").ejAutocomplete("open"); | **Method:** *showPopup*

@Html.EJS().Autocomplete("autocomplete").Render()

\$("#autocomplete").ejAutocomplete("showPopup"); |

| **Close event** | **Event:** *Close*

@Html.EJ().Autocomplete("autocomplete").close("onClose") | **Event:** *Close*

@Html.EJS().Autocomplete("autocomplete").Close("close").Render() |

| **Open event** | **Event:** *Open*

@Html.EJ().Autocomplete("autocomplete").open("onOpen") | **Event:** *Open*

@Html.EJS().Autocomplete("autocomplete").Open(open).Render() |

CSS

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property:** *Cssclass*

@Html.EJ().Autocomplete("autocomplete").CssClass("CSSClass") | **Property:** *cssClass*

@Html.EJS().Autocomplete("autocomplete").CssClass("class").Render() |

| **Height** | **Property:** *Height*
 @Html.EJ().Autocomplete("autocomplete").Height("300px") | By using *css-class* property, you can acheive this. |

| **showRoundedCorner** | **Property:** *ShowRoundedCorner*

@Html.EJ().Autocomplete("autocomplete").ShowRoundedCorner(true) | **By using *css-class* property, you can acheive this.** |

| **Width** | **Property:** *width*
 @Html.EJ().Autocomplete("autocomplete").Width("200px") |

Property: *width*
 @Html.EJS().Autocomplete("autocomplete").Width("300px").Render() |

| **Visibility** | **Property:** *visible*
 @Html.EJ().Autocomplete("autocomplete").Visible(true) | **By using *css-class* property, you can acheive this.** |

Grouping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property:** *fields.groupBy*

@Html.EJ().Autocomplete("autocomplete").AutocompleteFields(field =>

field.GroupBy("UniqueKey")) | **Property:** *fields.groupBy*

@Html.EJS().AutoComplete("games").Fields(new AutoCompleteFieldSettings { GroupBy = "Game" }).Render() |

Localization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property:** *Locale*
 @Html.EJ().Autocomplete("autocomplete").Locale("fr-FE") |

Property: *Locale*
 @Html.EJS().AutoComplete("games").Locale("fr-FE").Render() |

Template

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property:** *Template*

@Html.EJ().Autocomplete("autocomplete").Template("@Html.Raw("\${FirstName}\${City}"))" | **Property:** *ItemTemplate*

@Html.EJS().Autocomplete("autocomplete").ItemTemplate("@Html.Raw("\${FirstName}\${City}"))" |

| **Group Template** | **Not Applicable** | **Property:** *GroupTemplate*

@Html.EJS().Autocomplete("autocomplete").GroupTemplate("@Html.Raw("\${City}")) |

| **ValueTemplate** | **Not applicable** | **Property:** *ValueTemplate*

@Html.EJS().Autocomplete("autocomplete").ValueTemplate("data") |

| **Header Template** | **Not applicable** | **Property:** *HeaderTemplate*

 @Html.EJS().Autocomplete("autocomplete").HeaderTemplate("@Html.Raw("NameCity")) |

| **FooterTemplate** | **Not applicable** | **Property:** *FooterTemplate*

@Html.EJS().Autocomplete("autocomplete").FooterTemplate("@Html.Raw(" Total list items: " + 7 + "")) |

| **No records Template** | **Not applicable** | **Property:** *NoRecordsTemplate*

@Html.EJS().Autocomplete("autocomplete").NoRecordsTemplate("@Html.Raw(" NO DATA AVAILABLE")) |

| **Action failure Template** | **Not applicable** | **Property:** *ActionFailureTemplate*

@Html.EJS().Autocomplete("autocomplete").ActionFailureTemplate("@Html.Raw("<div class='header'> Photo Employee Info</div>")) |

Sorting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property:** *AllowSorting*

 @Html.EJ().Autocomplete("autocomplete").AllowSorting(true) | **It is applicable when the sortOrder is defined.** |

| **Order of sorting** | **Property:** *SortOrder*

@Html.EJS().Autocomplete("autocomplete").SortOrder("Ascending") | **Property:** *SortOrder*

 @Html.EJS().AutoComplete("games").SortOrder("Ascending").Render() |

Accessibility

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **RTL support** | **Property:** *EnableRtl*

@Html.EJ().Autocomplete("autocomplete").EnableRtl(true) | **Property:** *EnableRtl*

@Html.EJS().AutoComplete("games").EnableRtl(true).Render() |

Selection

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| ----- | ----- | ----- |

| **Selecting particular value** | **Property:** *SelectValueByKey*

@Html.EJ().Autocomplete("autocomplete").SelectValueByKey(12) | **Achievable through value property** |

| Selecting particular value | Property:

`value
@Html.EJ().Autocomplete("autocomplete").Vallue("data")` | **Property:** `value
@Html.EJS().AutoComplete("games").Value("data").Render()` |

| Selecting particular text | Property: `text
`

`@Html.EJ().Autocomplete("autocomplete").Text("data")` | **By using text property, you can acheive this.** |

| Selecting particular value | Method:

`SelectValueByKey
@Html.EJ().Autocomplete("autocomplete")

$("#autocomplete").selectValueByKey("key")` | **Achievable through value property** |

| Selecting particular text | Method: `SelectValueByText
`

`@Html.EJ().Autocomplete("autocomplete")

$("#autocomplete").selectValueByText("key")` | **Achievable through text property** |

| Select event | Event: `Select
@Html.EJ().Autocomplete("autocomplete").Select("onSelect")` |

Event: `Select
 @Html.EJS().AutoComplete("games").Select("onSelect").Render()` |

Miscellaneous**| Behavior | API in Essential JS 1 | API in Essential JS 2 |**

| --- | --- | --- |

| Enable/disable | Property:

`Enabled
@Html.EJ().AutoComplete("autocomplete").Enabled(true)` | **Property:** `Enabled
@Html.EJS().AutoComplete("games").Enabled(true).Render()` |

| Enable persistence | Property: `EnablePersistence
`

`@Html.EJ().AutoComplete("autocomplete").EnablePersistence(true)` | **Property:** `EnablePersistence
 @Html.EJS().AutoComplete("games").EnablePersistence(true).Render()` |

| Loading icon | Property: `ShowLoadingIcon`

`
@Html.EJ().AutoComplete("autocomplete").ShowLoadingIcon(true)` | **By default,it is showing** |

| Read only | Property: `ReadOnly
`

`@Html.EJ().AutoComplete("autocomplete").ReadOnly(true)` | **Property:** `ReadOnly @Html.EJS().AutoComplete("games").ReadOnly(true).Render()` |

| Disable | Method: `disable
`

`@Html.EJ().AutoComplete("autocomplete")

$("#autocomplete").ejAutoComplete("disable");` | **Enabled property can be used here** |

Common**| Behavior | API in Essential JS 1 | API in Essential JS 2 |**

| --- | --- | --- |

| Addition of new option watermarktext | Property: `AddNewText
`

`@Html.EJ().Autocomplete("autocomplete").AddNewText("data")` | **Not applicable** |

| **Addition of new item** | **Property:** *AllowAddNew*

```
<br/>@Html.EJ().Autocomplete("autocomplete").AllowAddNew(true) | Property:  
AllowCustom<br/> @Html.EJS().Autocomplete("autocomplete").AllowCustom(true).Render() |
```

| **Reset the autocomplete** | **Property:** *ShowResetIcon*

```
<br/>@Html.EJ().Autocomplete("autocomplete").ShowResetIcon(true) | Property: ShowClearIcon  
<br/> @Html.EJS().Autocomplete("autocomplete").ShowClearIcon(true).Render() |
```

| **Destroy** | **Method:** *destroy*


```
@Html.EJ().Autocomplete("autocomplete").AllowSorting(true)<br/><br/>$("#autocomplete").ej  
AutoComplete("destroy"); | Method: destroy  
<br/>@Html.EJS().Autocomplete("autocomplete").Render()<br/> <br/>var autoObj =  
document.getElementById("autocomplete").ej2_Instances[0];<br/><br/>autoObj.destroy(); |
```

| **Reset the autocomplete** | **Method:**

```
clearText<br/>@Html.EJ().Autocomplete("autocomplete")<br/><br/>$("#autocomplete").ejAutoC  
omplete("clearText"); | By passing empty value to the value property, you can acheive this |
```

| **Multicolumn** | **Property:** *MultiColumnSettings*


```
@Html.EJ().Autocomplete("autocomplete").Datasource((IEnumerable<CarsList>ViewBag.datas  
ource).<br/>MultiColumnSettings(obj => obj.Enable(true).Columns(obj1  
=>{obj1.Field("EmployeeID").HeaderText("EmployeeID").Add();obj1.Field("FirstName").Header  
Text("FirstName").Add();obj1.Field("City").HeaderText("City").Add();}).ShowHeader(true).Searc  
hColumnIndices(new List<int> { 0,1,2 }).StringFormat("{0}")) | Not applicable |
```

| **Hide the Autocomplete** | **Method:**

```
hide<br/><br/>@Html.EJ().Autocomplete("autocomplete")<br/><br/>$("#autocomplete").ejAuto  
Complete("hide"); | By using css-class property, you can acheive this.
```

| **Getting particular text** | **Method:** *GetActiveText*

```
<br/>@Html.EJ().Autocomplete("autocomplete")<br/><br/>$("#autocomplete").ejAutoComplete  
("getActiveText"); | By using text property, you can get it. |
```

| **Getting particular value** | **Method:** *getValue*


```
@Html.EJ().Autocomplete("autocomplete")<br/><br/>$("#autocomplete").ejAutoComplete("get  
Value"); | By using value property, you can get it. |
```

| **Change event** | **Event:**

```
Change<br/>@Html.EJ().Autocomplete("autocomplete").Change("onChange") | Event: Change  
<br/>@Html.EJS().Autocomplete("autocomplete").Change("onChange").Render() |
```

| **Create event** | **Event:** *Create*

```
<br/>@Html.EJ().Autocomplete("autocomplete").Created("onCreate") | Event: Created  
<br/>@Html.EJS().Autocomplete("autocomplete").Created("onCreate").Render() |
```

| **Destroy event** | **Event:** *Destroy*

```
<br/>@Html.EJ().Autocomplete("autocomplete").Destroy("onDestroy") | Event: Destroyed  
<br/>@Html.EJS().Autocomplete("autocomplete").Destroyed("ondestroy").Render() |
```

| Focus out event | Event:

FocusOut
@Html.EJ().Autocomplete("autocomplete").FocusOut("onFocusOut") | **Event:** *Blur*

@Html.EJS().Autocomplete("autocomplete").Blur("onBlur").Render() |

| Focus in event | Event :

FocusIn
@Html.EJ().Autocomplete("autocomplete").FocusIn("onFocus") | **Event:** *Focus*

@Html.EJS().Autocomplete("autocomplete").FocusIn("onFocus").Render() |

Avatar

Getting Started with ASP.NET MVC Avatar Control

This section briefly explains about how to include [ASP.NET MVC Avatar](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET web application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Add stylesheet

Here, the theme is referred using CDN inside the <head> of ~/Views/Shared/_Layout.cshtml file as follows,

~/_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{ {
site.ej2version } }/fluent.css" />
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls.

Add ASP.NET MVC Avatar control

Now, add the Syncfusion ASP.NET MVC Avatar control in ~/Home/Index.cshtml page.

CSHTML

```
<div id='element'>
  <span class="e-avatar e-avatar-xlarge"></span>
  <span class="e-avatar e-avatar-large"></span>
  <span class="e-avatar"></span>
  <span class="e-avatar e-avatar-small"></span>
  <span class="e-avatar e-avatar-xsmall"></span>
</div>
<style>
#element {
```

```
display: block;
width: 300px;
margin: 130px auto;
border-radius: 3px;
justify-content: center;
}
.e-avatar {
background-image:
url(@Url.Content("~/Content/avatar/images/pic01.png"));
margin: 2px;
}
</style>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Avatar control will be rendered in the default web browser.



Note: [View Sample in GitHub.](#)

See also
[Types of Avatar](#)

Types and Styles in ASP.NET MVC Avatar Control

This section explains different types of avatar.

Avatar size

The Essential JS 2 Avatar has the following predefined sizes that can be used with the `.e-avatar` class to change the appearance of the avatar.

Class Name	Description
-----	-----
e-avatar-xlarge	Displays xlarge size avatar.
e-avatar-large	Displays apply large size avatar.
e-avatar	Displays apply default size avatar.
e-avatar-small	Displays apply small size avatar.
e-avatar-xsmall	Displays apply xsmall size avatar.

CSHTML

```
<div id='element'>
  <span class="e-avatar e-avatar-xlarge"></span>
  <span class="e-avatar e-avatar-large"></span>
  <span class="e-avatar"></span>
  <span class="e-avatar e-avatar-small"></span>
  <span class="e-avatar e-avatar-xsmall"></span>
```

```

</div>
<style>
#element {
    display: block;
    width: 300px;
    margin: 100px auto;
    border-radius: 3px;
}
.e-avatar {
    background-image: url(../pic01.png);
}
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```



Avatar types

The types of Essential JS 2 avatar are:

- Default
- Circle

Default

The default style of the avatar is rectangular shape with rounded corners, which can be applied from adding the modifier class `.e-avatar` to the target element.

CSHTML

```

<div id='element'>
    <span class="e-avatar e-avatar-xlarge">RT</span>
    <span class="e-avatar e-avatar-large">RT</span>

```



```

<span class="e-avatar">RT</span>
<span class="e-avatar e-avatar-small">RT</span>
<span class="e-avatar e-avatar-xsmall">RT</span>
</div>
<style>
#element {
    display: block;
    width: 260px;
    margin: 100px auto;
    border-radius: 3px;
}
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```



Circle

The circle avatar style can be applied by adding the modifier class `e-avatar-circle` to the target element.

CSHTML

```

<div id='element'>
    <span class="e-avatar e-avatar-xlarge e-avatar-circle">SJ</span>
    <span class="e-avatar e-avatar-large e-avatar-circle">SJ</span>
    <span class="e-avatar e-avatar-circle">SJ</span>
    <span class="e-avatar e-avatar-small e-avatar-circle">SJ</span>
    <span class="e-avatar e-avatar-xsmall e-avatar-circle">SJ</span>
</div>
<style>
#element {
    display: block;
    width: 300px;
}

```

```
margin: 100px auto;
border-radius: 3px;
}
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



Note: [View Sample in GitHub.](#)

How To

Avatar Customization

Colour customization

The avatar comes with default background colour (grey). This can be easily customized to desired colour by adding custom class or directly selecting the avatar class from the CSS.

CSHTML

```
<div id='element'>
    <span class="e-avatar e-avatar-xlarge e-avatar-circle green">AJ</span>
    <span class="e-avatar e-avatar-xlarge e-avatar-circle violet">JK</span>
    <span class="e-avatar e-avatar-xlarge e-avatar-circle rose">EL</span>
    <span class="e-avatar e-avatar-xlarge e-avatar-circle blue">SR</span>
    <span class="e-avatar e-avatar-xlarge e-avatar-circle red">PD</span>
</div>
<style>
#element {
    display: flex;
    width: 400px;
    margin: 100px auto;
    border-radius: 3px;
    justify-content: center;
}
```

```
.e-avatar {
    margin: 2px;
}
.e-avatar.green {
    background-color: #87eb87;
}
.e-avatar.violet {
    background-color: #ee82ee;
}
.e-avatar.blue {
    background-color: #7171e4;
}
.e-avatar.red {
    background-color: #d86e6e;
}
.e-avatar.rose {
    background-color: #bc8f8f;
}
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



Customize avatar sizes

Even though the avatar comes with five predefined sizes, sometimes it's not enough. So, the avatar is designed in such a way that the width and height will be relative to font-size. By changing the font-size of the avatar element, you can change the width and height automatically.

CSHTML

```
<div id='element'>
    <span class="e-avatar">26px</span>
    <span class="e-avatar">24px</span>
```

```
<span class="e-avatar">22px</span>
<span class="e-avatar">20px</span>
<span class="e-avatar">18px</span>
</div>
<style>
#element {
    display: block;
    width: 400px;
    margin: 100px auto;
    border-radius: 3px;
}
#element :nth-child(5) {
    font-size: 18px;
}
#element :nth-child(4) {
    font-size: 20px;
}
#element :nth-child(3) {
    font-size: 22px;
}
#element :nth-child(2) {
    font-size: 24px;
}
#element :nth-child(1) {
    font-size: 26px;
}
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



Note: [View Sample in GitHub.](#)

Use various media in avatar

Avatars can be used with a wide variety of media formats like SVG, font-icons, images, letters, words, etc.

CSHTML

```
<div class="sample_container avatar-types">
  <div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
      <div class="e-card-content">
        <!-- XLarge Circle Avatar Component -->
        <div class="e-avatar e-avatar-xlarge e-avatar-circle
image"></div>
      </div>
      <div class="e-card-content">
        <div>Image</div>
      </div>
    </div>
  </div>
  <div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
      <div class="e-card-content">
        <!-- XLarge Circle Avatar Component -->
        <div class="e-avatar e-avatar-xlarge e-avatar-circle">
          <div class="svg_icons chrome"></div>
        </div>
      </div>
      <div class="e-card-content">
        <div>SVG</div>
      </div>
    </div>
  </div>
  <div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
      <div class="e-card-content">
        <!-- XLarge Circle Avatar Component -->
        <div class="e-avatar e-avatar-xlarge e-avatar-
circle">GR</div>
      </div>
      <div class="e-card-content">
        <div>Initial</div>
      </div>
    </div>
  </div>
  <div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
      <div class="e-card-content">
        <!-- XLarge Circle Avatar Component -->
        <div class="e-avatar e-avatar-xlarge e-avatar-circle">
          <div class="e-people_icons"></div>
        </div>
      </div>
      <div class="e-card-content">
```

```

        <div>FontIcon</div>
    </div>
</div>
<div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <!-- XLarge Circle Avatar Component -->
            <div class="e-avatar e-avatar-xlarge e-avatar-
circle">User</div>
        </div>
        <div class="e-card-content">
            <div>Word</div>
        </div>
    </div>
</div>
<div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <!-- XLarge Circle Avatar Component -->
            <div class="e-avatar e-avatar-xlarge e-avatar-circle
custom">
                <div class="e-people icons"></div>
            </div>
            <div class="e-card-content">
                <div>Custom</div>
            </div>
        </div>
    </div>
</div>
<style>
/* Media Quries for various devices */
@media only screen and (max-width: 965px) {
    .sample_container.avatar-types {
        max-width: 265px;
        margin: auto;
        margin-top: 0;
    }
    .e-avatar-showcase.e-card {
        width: 120px;
    }
}
@media only screen and (min-width: 965px) {
    .sample_container.avatar-types {
        max-width: 488px;
        margin: auto;
        margin-top: 35px;
    }
    .e-avatar-showcase.e-card {
        width: 150px;
    }
}
.sample_container.avatar-types .avatar-block {
    display: inline-block;

```

```

        vertical-align: top;
    }
    /* Avatar image source in 'background-image' property */
    .e-avatar.image {
        background-image: url(./pic01.png);
        background-repeat: no-repeat;
        background-size: cover;
        background-position: center;
    }
    /* SVG Icons */
    .chrome {
        background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%23ffffff' d='M16.033 11.049a5.155 5.155 0 1 1 0 10.312 5.156 5.156 0
0 1 0-10.312zM16.124 0c1.281-.003 9.659.318 14.268 9.043h-
.016l.01.018c.33.578 3.745 6.94-.485 14.969 0 0-4.215 8.107-14.565 7.968l-
.452-.012-.004.007-.004.007.02-.037c.564-.98 5.112-8.884 6.357-11.067l.016-
.028.007-.008.04-.069.11-.127a7.085 7.085 0 0 0 1.457-2.967l.01-.046.035-
.151c.088-.424.148-.944.128-1.549l-.006-.117v-.004l-.007-.143-.006-.07-.005-
.079-.012-.116v-.011-.001-.008-.016-.158a7.2 7.2 0 0 0-.096-.572l-.018-.081-
.013-.064a9.801 9.801 0 0 0-.692-2.016c-.165-.243-.332-.489-.512-.733l-.142-
.187 8.728-2.554s-10.538-.01-13.018-.001l.021.005H16.642l-.14-.013a7.034
7.034 0 0 0-1.132-.003l-.167.016h-.047l-.034-.001c-.193.002-1.213.045-
2.492.764l-.005.003-.033.016a7.158 7.158 0 0 0-3.25 3.533l-.059.148-6.485-
6.404s4.74 8.311 6.165 10.779l.065.113.023.088a7.14 7.14 0 0 0 7.777
5.118l.144-.02L14.854 32h-.027c-.667-.005-7.894-.234-12.744-7.906 0 0-4.925-
7.698.37-16.573l.252-.411.001-.002C2.822 6.904 6.58.374 15.958.003c0 0 .057-
.003.166-.003z'/%3E%3C/svg%3E") no-repeat 100% 100%;
    }
    .svg_icons {
        width: 32px;
        height: 32px;
        display: inline-block;
    }
    /* Card Customization */
    .e-avatar-showcase.e-card {
        height: 113px;
        padding: 5px;
        margin: 5px;
        box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12), 0 1px 2px rgba(0, 0, 0,
0.24);
        border-radius: 8px;
    }
    .e-avatar-showcase.e-card .e-card-header .e-card-header-title {
        align-self: center;
        font-size: 18px;
        letter-spacing: 1px;
        text-shadow: #eaeaea 1px 1px 2px;
    }
    .e-avatar-showcase.e-card .e-card-header .e-card-sub-title {
        color: rgba(0, 0, 0, 0.75);
        white-space: pre-line;
        font-size: 14px;
        text-shadow: #eaeaea 1px 1px 2px;
    }
    .e-avatar-showcase.e-card .e-card-header .e-card-sub-title p {
        margin: 0;
    }

```

```

}
.e-avatar-showcase.e-card .e-card-content {
    align-self: center;
    padding: 10px 0 10px 0;
    overflow: visible;
}
.e-avatar-showcase.e-card .e-card-content .e-avatar {
    font-size: 18px;
}
/* Font Icons */
@font-face {
    font-family: 'Contacts';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRgAAAEoAAAAVmNtYXNnEODaAAABjAAAADhnbHlmiAn
WagAAAcwAAADwaGVhZBD04psAAADQAAAAANmhoZWEHiwNuAAAArAAAACRobXR4C9AAAAAAAYAAAA
MbG9jYQAwAHgAAAEHAAACG1heHABDwA1AAABCAAAACBuYW1lby+ImAAAArWAAAIxcG9zdGUyI4A
AAATwAAAAOwABAAADUv9qAFoEAAAAAAD3QABAAAAAAAAAAAAAAAAAAwABAAAAQAAWW9ja18
PPPUACwPoAAAAANb8zuYAAAAA1vzO5gAAAAAD3QPqAAAACAACAAAAAAAAAAAEAAAADACkAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQpWAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAAAAAAAgAAAMAAAAUAAMAAQAAABQABAakAAAABAAEAAEAOCB//8AAOCa//8AAAABAAQAAAA
BAAIAAAAAADAAeAACAAAAAO+A+oADQAZAAA3FBYXIT4BNS4BJyEOARMeARc+ATcuAScOAS4YFwM
zFxfGd4H+zYGr4QOCY2KCAwOCYmGCnCtOISFOK3qiAwOiAeligwICg2JjggICggAAAAACAAAAAP
dA+oAFAAoAAABDgEHIicjDgEHLgEnLgEnPgE3HgEFFBYfARYfATcXFhc2JDcmJCCGBAOkBfK3Kio
XEFcpBBEMRUsBBfK3tvL8cVRLDggBBsQKLDDPARMFbf7tz87+7QI+ndEEBwI/Izh2DS+RVZ3RBAT
RnWCmN3BIETecAgcBBPK1tfIEBPIAAAAABIA3gABAAAAAAAAAAEAAAABAAAAAABAAgAAQABAAA
AAAACAACACQABAAAAAADAAGAEABAAAAAAAEAAgAGAABAAAAAAFAAsAIAABAAAAAAGAAGAKwA
BAAAAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4
AgwADAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQ
AFAFgA1wADAAEECQALACQBLyBDb250YWN0c1JlZ3VsYXJDb250YWN0c0NvbnRhY3RzVmVyc2lvbiA
xLjBDb250YWN0c0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvZ3d
3LnN5bmNmdXNpb24uY29tACAAQwBvAG4AdABhAGMAdABzAFIAZQBnAHUAbABhAHIAQwBvAG4AdAB
hAGMAdABzAEMAbwBuAHQAYQBjAHQAQwBvAG4AdABhAGMAdABzAEYAbwBuAHQAIABnAGUAbgBlAHIA
YQB0AGUAZAAGAHUAQwBvAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIA
BTAAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAA
AAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQ
ABHVzZXIKY2hhdc0wMS13ZgAAAA==) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.people,
.e-people {
    font-family: 'Contacts';
}
.e-people:before {
    content: '\e700';
}
.e-avatar .e-people.icons {
    font-size: 24px;
}
/* Custom Avatar Background Color */
.e-avatar.custom {
    background-color: blueviolet;
}
</style>

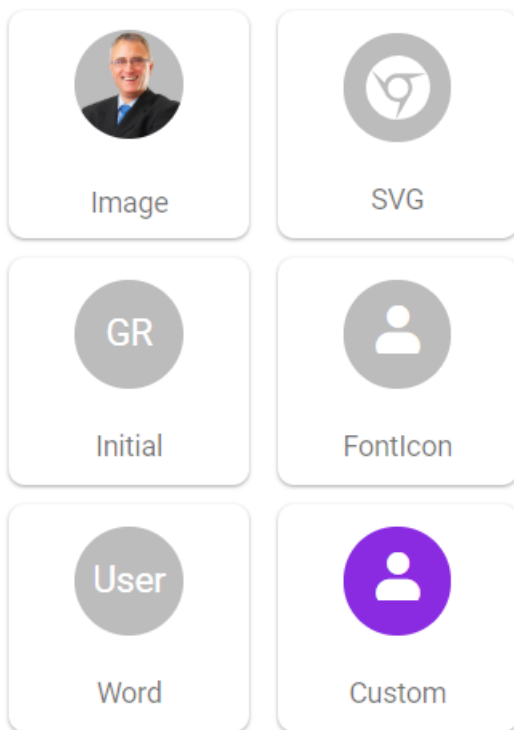
```


HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```

**Integrate avatar into ListView**

Avatar is integrated into the listview to create contacts applications. The `xsmall` size avatar is used to match the size of the list item. Letters and images are also used as avatar content.

CSHTML

```

@using Syncfusion.EJ2
<div class="col-lg-12 control-section">
    <div class="sample_container">
        <!-- Listview element -->
        @{ var template = "<div class='listWrapper'>" +
            "${if (avatar != '')}" +

```

```

        "<span class='e-avatar e-avatar-small e-avatar-
circle'>${avatar}</span>" +
        "${else}" +
        "<span class='${pic} e-avatar e-avatar-small e-avatar-circle'>
</span>" +
        "${/if}" +
        "<span class='text'>" +
        "${text} </span> </div>";}

@Html.EJS().ListView("letterAvatarList").Enable(true).DataSource((IEnumerable<
<b>object</b>>)ViewBag.dataSource).HeaderTitle("Contacts").ShowHeader(true).Templ
ate(template).SortOrder(Syncfusion.EJ2.Lists.SortOrder.Ascending).Render()
</div>
</div>
<style>
    .control-section {
        overflow: auto;
    }
    /* ListView Customization */
    #letterAvatarList {
        max-width: 350px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #letterAvatarList .listWrapper {
        width: inherit;
        height: inherit;
        position: relative;
    }
    #letterAvatarList .e-list-header {
        height: 54px;
    }
    .material #letterAvatarList .e-list-header .e-text {
        line-height: 22px;
    }
    .fabric #letterAvatarList .e-list-header .e-text {
        line-height: 22px;
    }
    .bootstrap #letterAvatarList .e-list-header .e-text {
        line-height: 13px;
    }
    .highcontrast #letterAvatarList .e-list-header .e-text {
        line-height: 20px;
    }
    #letterAvatarList .e-list-item {
        cursor: pointer;
        height: 50px;
        line-height: 44px;
        border: 0;
    }
    /* Badge Positioning */
    #letterAvatarList .e-badge {
        margin-top: 12px;
    }
    #letterAvatarList .listWrapper .text {
        width: 60%;

```

```

        display: inline-block;
        vertical-align: middle;
        margin: 0 50px;
    }
    /* Avatar Positioning */
    #letterAvatarList .listWrapper .e-avatar {
        position: absolute;
        top: calc(100% - 40px);
        font-size: 10px;
        left: 5px;
    }
    /* Avatar Background Customization */
    #letterAvatarList .e-list-item:nth-child(1) .e-avatar {
        background-color: #039BE5;
    }
    #letterAvatarList .e-list-item:nth-child(3) .e-avatar {
        background-color: #E91E63;
    }
    #letterAvatarList .e-list-item:nth-child(5) .e-avatar {
        background-color: #009688;
    }
    /* Avatar images using 'background-image' property */
    .pic01 {
        background-image: url('./pic01.png');
    }
    .pic02 {
        background-image: url('./pic02.png');
    }
    .pic03 {
        background-image: url('./pic03.png');
    }
    .pic04 {
        background-image: url('./pic04.png');
    }
}
</style>

```

HOMECONTROLLER.CS

```

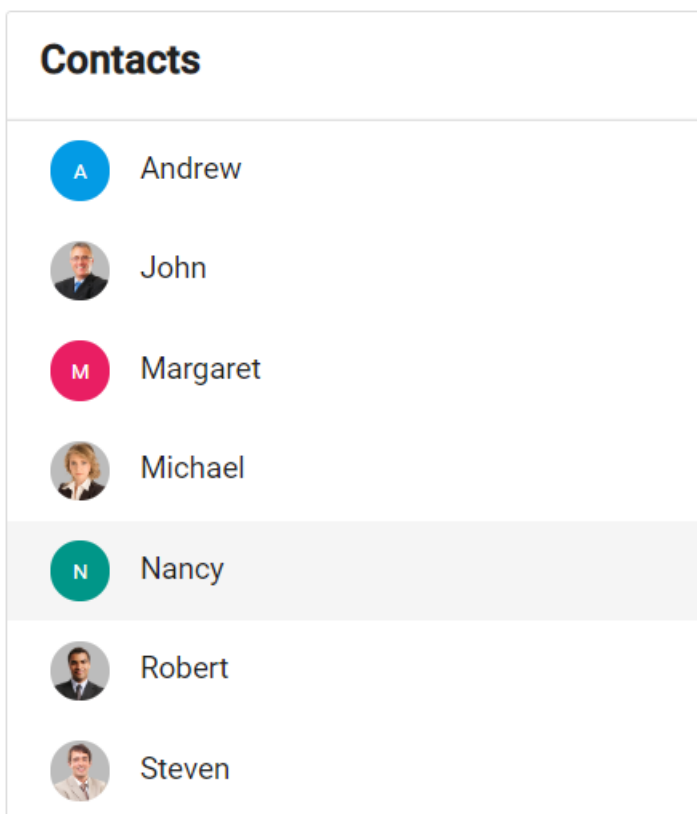
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers.Avatar
{
    public partial class AvatarController : Controller
    {
        public ActionResult Listview()
        {
            List<object> data = new List<object>();
            data.Add(new { text = "Robert", id = "s_01", avatar = "", pic=
"pic04" });
            data.Add(new { text = "Nancy", id = "s_02", avatar = "N", pic= ""
});
            data.Add(new { text = "John", id = "s_03", pic= "pic01", avatar=
"" });

```

```

        data.Add(new { text = "Andrew", id = "s_04", avatar= "A", pic=
"" });
        data.Add(new { text = "Michael", id = "s_05", pic= "pic02",
avatar= "" });
        data.Add(new { text = "Steven", id = "s_06", pic= "pic03",
avatar= "" });
        data.Add(new { text = "Margaret", id = "s_07", avatar= "M", pic=
"" });
        ViewBag.dataSource = data;
        return View();
    }
}

```



Note: [View Sample in GitHub.](#)

Integrate avatar into Badge

The badge is dependent and supportive component, and it can be used with avatar to create a notification avatar. The default avatar (.e-avatar) and circle avatar (.e-avatar-circle) have been used with notification badges (.e-badge-notification) in the following sample.

CSHTML

```

<div class="sample_container avatar-badge">
    <div class="avatar-block">
        <!-- Card Component -->
        <div class="e-card e-avatar-showcase">

```

```

        <div class="e-card-content">
            <div class="avatar-sub-block">
                <!-- xSmall Avatar-->
                <div class="e-avatar e-avatar-xsmall">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification e-badge-circle">6</span>
            </div>
            <div class="avatar-sub-block">
                <!-- Small Avatar-->
                <div class="e-avatar e-avatar-small">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification e-badge-circle">12</span>
            </div>
            <div class="avatar-sub-block">
                <!-- Avatar-->
                <div class="e-avatar">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification">46</span>
            </div>
            <div class="avatar-sub-block">
                <!-- Large Avatar-->
                <div class="e-avatar e-avatar-large">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification">82</span>
            </div>
            <div class="avatar-sub-block">
                <!-- xLarge Avatar-->
                <div class="e-avatar e-avatar-xlarge">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification">99+</span>
            </div>
        </div>
    </div>
    <div class="circleAvatar avatar-block">
        <!-- Card Component -->
        <div class="e-card e-avatar-showcase">
            <div class="e-card-content">
                <div class="avatar-sub-block">
                    <!-- xSmall Circle Avatar-->
                    <div class="e-avatar e-avatar-circle e-avatar-xsmall">
                        
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
        <!-- Notification Badge -->
        <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification e-badge-circle">6</span>
    </div>
    <div class="avatar-sub-block">
        <!-- Small Circle Avatar-->
        <div class="e-avatar e-avatar-circle e-avatar-small">
            
        </div>
        <!-- Notification Badge -->
        <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification e-badge-circle">12</span>
    </div>
    <div class="avatar-sub-block">
        <!-- Circle Avatar-->
        <div class="e-avatar e-avatar-circle">
            
        </div>
        <!-- Notification Badge -->
        <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification">46</span>
    </div>
    <div class="avatar-sub-block">
        <!-- Large Circle Avatar-->
        <div class="e-avatar e-avatar-circle e-avatar-large">
            
        </div>
        <!-- Notification Badge -->
        <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification">82</span>
    </div>
    <div class="avatar-sub-block">
        <!-- xLarge Circle Avatar-->
        <div class="e-avatar e-avatar-circle e-avatar-xlarge">
            
        </div>
        <!-- Notification Badge -->
        <span class="e-badge e-badge-primary e-badge-overlap e-
badge-notification">99+</span>
    </div>
</div>
</div>
</div>
</div>
<style>
.sample_container.avatar-badge .avatar-sub-block {
    display: inline-block;
    position: relative;
    margin: 7px
}
.sample_container.avatar-badge .avatar-block {
    display: inline-block;
    vertical-align: top;
}
/* Media Queries for various devices */
@@media only screen and (max-width: 965px) {

```

```

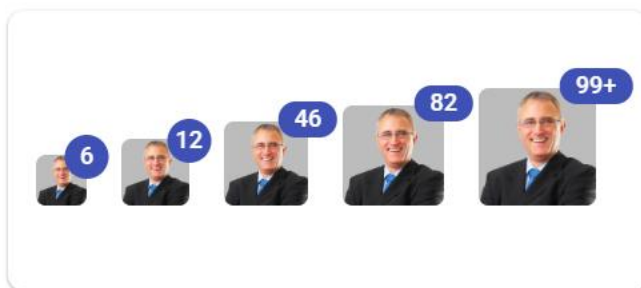
        .sample_container.avatar-badge {
            max-width: 332px;
            margin: auto;
            margin-top: 0;
        }
        .circleAvatar {
            margin-top: 15px;
        }
        .e-avatar-showcase.e-card {
            width: 320px;
        }
    }
    @@media only screen and (min-width: 965px) {
        .sample_container.avatar-badge {
            max-width: 825px;
            margin: auto;
            margin-top: 70px;
        }
        .e-avatar-showcase.e-card {
            width: 400px;
        }
    }
    .e-avatar.image {
        background-repeat: no-repeat;
        background-size: cover;
        background-position: center;
    }
    /* Card Customization */
    .e-avatar-showcase.e-card {
        height: 140px;
        padding: 4px;
        margin: 5px;
        box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12), 0 1px 2px rgba(0, 0, 0,
0.24);
        border-radius: 8px;
    }
    .e-avatar-showcase.e-card .e-card-header .e-card-header-title {
        align-self: center;
        font-size: 18px;
        letter-spacing: 1px;
        text-shadow: #eaeaea 1px 1px 2px;
    }
    .e-avatar-showcase.e-card .e-card-header .e-card-sub-title {
        color: rgba(0, 0, 0, 0.75);
        white-space: pre-line;
        font-size: 14px;
        text-shadow: #eaeaea 1px 1px 2px;
    }
    .e-avatar-showcase.e-card .e-card-header .e-card-sub-title p {
        margin: 0;
    }
    .e-avatar-showcase.e-card .e-card-content {
        align-self: center;
        padding: 10px 11px 10px 0px;
        color: rgba(0, 0, 0, 0.75);
        overflow: visible;
    }
}

```

```
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



Note: [View Sample in GitHub.](#)

Badge

Getting Started with ASP.NET MVC Badge Control

This section briefly explains about how to include [ASP.NET MVC Badge](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET web application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Add stylesheet

Here, the theme is referred using CDN inside the `<head>` of `~/Views/Shared/_Layout.cshtml` file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls.

Add ASP.NET MVC Badge control

Now, add the Syncfusion ASP.NET MVC Badge control in `~/Home/Index.cshtml` page.

CSHTML

```
<div id='element'>
    <h1>Badge Component <span class="e-badge e-badge-
primary">New</span></h1>
</div>
<style>
#element {
    display: flex;
    width: 400px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
    justify-content: center;
}
</style>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Badge control will be rendered in the default web browser.

Badge Component New

Note: [View Sample in GitHub.](#)

See also

[Types of Badge](#)

Types in ASP.NET MVC Badge Control

This section explains different styles and types of the badges.

Badge styles

The Essential JS 2 Badge has the following predefined styles that can be used with `.e-badge` class to change the appearance of a badge.

Class Name	Description
----- -----	
e-badge-primary	Represents a primary notification.
e-badge-secondary	Represents a secondary notification.
e-badge-success	Represents a positive notification.
e-badge-danger	Represents a negative notification.
e-badge-warning	Represents notification with caution.
e-badge-info	Represents an informative notification.
e-badge-light	Represents notification in light variant.
e-badge-dark	Represents notification in dark variant.

CSHTML

```
<div id='element'>
  <div class="sample_container">
    <div class="block" style="display:inline-block">
      <div class="e-card e-badge-showcase">
        <div class="e-card-content">
          <div>
            <span class="e-badge e-badge-primary">Primary</span>
          </div>
        </div>
        <div class="e-card-content">
          <div>
            <code>.e-badge-primary</code>
          </div>
        </div>
      </div>
    </div>
    <div class="block" style="display:inline-block">
      <div class="e-card e-badge-showcase">
        <div class="e-card-content">
          <span class="e-badge e-badge-secondary">Secondary</span>
        </div>
        <div class="e-card-content">
          <code>.e-badge-secondary</code>
        </div>
      </div>
    </div>
    <div class="block" style="display:inline-block">
```

```

        <div class="e-card e-badge-showcase">
            <div class="e-card-content">
                <span class="e-badge e-badge-success">Success</span>
            </div>
            <div class="e-card-content">
                <code>.e-badge-success</code>
            </div>
        </div>
    </div>
    <div class="block" style="display:inline-block">
        <div class="e-card e-badge-showcase">
            <div class="e-card-content">
                <span class="e-badge e-badge-danger">Danger</span>
            </div>
            <div class="e-card-content">
                <code>.e-badge-danger</code>
            </div>
        </div>
    </div>
    <div class="block" style="display:inline-block">
        <div class="e-card e-badge-showcase">
            <div class="e-card-content">
                <span class="e-badge e-badge-warning">Warning</span>
            </div>
            <div class="e-card-content">
                <code>.e-badge-warning</code>
            </div>
        </div>
    </div>
    <div class="block" style="display:inline-block">
        <div class="e-card e-badge-showcase">
            <div class="e-card-content">
                <span class="e-badge e-badge-info">Info</span>
            </div>
            <div class="e-card-content">
                <code>.e-badge-info</code>
            </div>
        </div>
    </div>
    <div class="block" style="display:inline-block">
        <div class="e-card e-badge-showcase">
            <div class="e-card-content">
                <span class="e-badge e-badge-light">Light</span>
            </div>
            <div class="e-card-content">
                <code>.e-badge-light</code>
            </div>
        </div>
    </div>
    <div class="block" style="display:inline-block">
        <div class="e-card e-badge-showcase">
            <div class="e-card-content">
                <span class="e-badge e-badge-dark">Dark</span>
            </div>
            <div class="e-card-content">
                <code>.e-badge-dark</code>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
</div>
<Style>
#element {
    width: 100%;
    margin: auto;
}
.sample_container {
    width: 800px;
    margin: auto;
}
.block {
    margin: 10px 10px 10px 10px;
    vertical-align: top;
}
.e-badge-showcase.e-card {
    width: 150px;
    height: 90px;
    padding: 5px;
    margin: 5px;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12), 0 1px 2px rgba(0, 0, 0,
0.24);
    border-radius: 8px;
}
.e-badge-showcase.e-card .e-card-header .e-card-header-title {
    align-self: center;
    font-size: 18px;
    letter-spacing: 1px;
    text-shadow: #eaeaea 1px 1px 2px;
}
.e-badge-showcase.e-card .e-card-header .e-card-sub-title {
    color: rgba(0, 0, 0, 0.75);
    white-space: pre-line;
    font-size: 14px;
    text-shadow: #eaeaea 1px 1px 2px
}
.e-badge-showcase.e-card .e-card-header .e-card-sub-title p {
    margin: 0;
}
.e-badge-showcase.e-card .e-card-content {
    align-self: center;
    padding: 0 0 10px 0;
}
.e-badge-showcase.e-card .e-card-content .e-badge {
    font-size: 12px;
}
</Style>

```

HOMECONTROLLER.CS

```

public ActionResult Badge()
{
    return View();
}

```



Badge types

The types of Essential JS 2 badges are as follows:

- Circle
- Pill
- Link
- Notification
- Overlap
- Dot
- Position

Circle

The circle badge style can be applied by adding the modifier class `.e-badge-circle` to the target element.

CSHTML

```
<div id='element'>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="skype svg_icons"></div>
    <span class="e-badge e-badge-success e-badge-overlap e-badge-notification e-badge-circle">18</span>
  </div>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="twitter svg_icons"></div>
    <span class="e-badge e-badge-info e-badge-overlap e-badge-notification e-badge-circle">9</span>
  </div>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="facebook svg_icons"></div>
    <span class="e-badge e-badge-info e-badge-overlap e-badge-notification e-badge-circle">2</span>
  </div>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="firefox svg_icons"></div>
```

```

    <span class="e-badge e-badge-danger e-badge-overlap e-badge-
notification e-badge-circle">35</span>
  </div>
</div>
<style>
#element {
  display: flex;
  width: 400px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
  justify-content: center;
  position: relative;
  top: 130px;
}
.facebook {
  background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%233c5a99;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M22.53,8.7h2.58V4.64H21.24a5.25,5.25,0,0,0-4,2s-1.06,1-
1.08,3.92h0v3H12.36v4.31h3.82V29h4.41V17.86h3.81.53-4.31H20.59v-
3h0A1.78,1.78,0,0,1,22.53,8.7Z'/%3e%3c/svg%3e");
}
.skype {
  background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%2331c4ed;%7d.cls-
2%7bfill:%23fff;fill-
rule:evenodd;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2' d='M19.5,19.14a4,4,0,0,1-
1.75,1.31,7.22,7.22,0,0,1-
2.68,4.6A6.6,6.6,0,0,1,12,20.27,4,4,0,0,1,10.58,19,2.71,2.71,0,0,1,10,17.51a1
.05,1.05,0,0,1,.36-.8,1.28,1.28,0,0,1,.9-
.33,1.14,1.14,0,0,1,.75,2.6,1.89,1.89,0,0,1,.51,7.3,4.54,4.54,0,0,0,.49,8.6,2,2
,0,0,0,.72,5.5A3.08,3.08,0,0,0,15,19a3,3,0,0,0,1.73-.45,1.23,1.23,0,0,0,.63-
1.06,1,1,0,0,0-.33-.8,2.3,2.3,0,0,0-.92-.49c-.39-.12-.92-.26-1.57-
.39a12.2,12.2,0,0,1-2.25-.67,3.64,3.64,0,0,1-1.48-1.06,2.47,2.47,0,0,1-.29-
.47,2.84,2.84,0,0,1-.26-1.22,2.71,2.71,0,0,1,.58-1.72,3.71,3.71,0,0,1,1.67-
1.14A7.37,7.37,0,0,1,15,9.14a7,7,0,0,1,2,.26,4.38,4.38,0,0,1,1.42,7,3.1,3.1,
0,0,1,.84,9.2,2.11,2.11,0,0,1,.26,1,1.13,1.13,0,0,1-.35,8.2,1.18,1.18,0,0,1-
.88,3.6,1,0,9,1,0,9,0,0,1-.74-.23,2.68,2.68,0,0,1-.51-.67,3,3,0,0,0-.77-
.94A2.42,2.42,0,0,0,14.87,11a2.76,2.76,0,0,0-1.49,3.6,1,1,0,0,0-
.53,8.1,7.8,7.8,0,0,0,.17,5,1.63,1.63,0,0,0,.52,3.8,4.48,4.48,0,0,0,.69,2.7,1.15,0
,1,.24c.69,1.5,1.33,3.2,1.89,4.9a6.29,6.29,0,0,1,1.47,6.7,2.81,2.81,0,0,1,1,1,3,
3,0,0,1,.35,1.49,3.15,3.15,0,0,1-.6,1.89Zm4-2.23a7.73,7.73,0,0,0,.2-1.81c0-
.22,0-.43,0-.64a8.53,8.53,0,0,0-8.56-7.83,8.72,8.72,0,0,0-
1.46,1.2A5.08,5.08,0,0,0,11,6a5,5,0,0,0-
5,4.91,4.83,4.83,0,0,0,.68,2.48,7.33,7.33,0,0,0-
.13,9.4,6.51,6.51,0,0,0,0,.77,8.52,8.52,0,0,0,8.58,8.46,8.9,8.9,0,0,0,1.57-
.14A5.09,5.09,0,0,0,19,24a4.94,4.94,0,0,0,5-4.91,4.86,4.86,0,0,0-.52-
2.18Z'/%3e%3c/svg%3e");
}
.twitter {

```

```

background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.firefox {
background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.svg_icons {
width: 32px;
height: 32px;
display: inline-block;
}
</style>

```

HOMECONTROLLER.CS

```

public ActionResult Badge()
{
    return View();
}

```



Pill

The pill badge style can be applied by adding the modifier class `.e-badge-pill` to the target element.

CSHTML

```

<div id='element'>
    <h1>Badge Component <span class="e-badge e-badge-primary e-badge-
pill">New</span></h1>

```

```
</div>
```

HOMECONTROLLER.CS

```
public ActionResult Badge()  
{  
    return View();  
}
```

Badge Component New

Link

When badge modifier classes are applied to the anchor tag, the badge's appearance will change from normal state to hover state on mouseover.

CSHTML

```
<div id='element'>  
    <div style="display: inline-block; margin-top: 15px;">  
        <a href="#" class="e-badge e-badge-primary">Link Badge</a>  
    </div>  
</div>  
<style>  
#element {  
    display: flex;  
    width: 400px;  
    height: 50px;  
    margin: auto;  
    border: 1px solid #dddddd;  
    border-radius: 3px;  
    justify-content: center;  
    position: relative;  
    top: 130px;  
}  
</style>
```

HOMECONTROLLER.CS

```
public ActionResult Badge()  
{  
    return View();  
}
```

Link Badge

Notification

The notification badge style can be applied by adding the modifier class `.e-badge-notification` to the target element. Notification badges are used when a content or a context needs special attention. While using the notification badge, set the parent element to `position: relative`.

CSHTML

```

<div id='element'>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="skype svg_icons"></div>
    <span class="e-badge e-badge-success e-badge-overlap e-badge-notification">99+</span>
  </div>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="twitter svg_icons"></div>
    <span class="e-badge e-badge-info e-badge-overlap e-badge-notification">27</span>
  </div>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="facebook svg_icons"></div>
    <span class="e-badge e-badge-info e-badge-overlap e-badge-notification">2</span>
  </div>
  <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="firefox svg_icons"></div>
    <span class="e-badge e-badge-danger e-badge-overlap e-badge-notification">35</span>
  </div>
</div>
<Style>
#element {
  display: flex;
  width: 400px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
  justify-content: center;
  position: relative;
  top: 130px;
}
.facebook {
  background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%233c5a99;%7d.cls-2%7bfill:%23fff;%7d%3c/style%3e%3cdefs%3e%3ctitle%3eIcon%3c/title%3e%3crect class='cls-1' x='1' y='1' width='28' height='28' rx='5.75' ry='5.75'/%3e%3cpath class='cls-2' d='M22.53,8.7h2.58V4.64H21.24a5.25,5.25,0,0,0-4,2s-1.06,1-1.08,3.92h0v3H12.36v4.31h3.82V29h4.41V17.86h3.81.53-4.31H20.59v-3h0A1.78,1.78,0,0,1,22.53,8.7Z'/%3e%3c/svg%3e");
}
.skype {
  background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%2331c4ed;%7d.cls-2%7bfill:%23fff;fill-rule:evenodd;%7d%3c/style%3e%3cdefs%3e%3ctitle%3eIcon%3c/title%3e%3crect class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'

```

```

ry='5.75'/%3e%3cpath class='cls-2' d='M19.5,19.14a4,4,0,0,1-
1.75,1.31,7.22,7.22,0,0,1-
2.68.46A6.6,6.6,0,0,1,12,20.27,4,4,0,0,1,10.58,19,2.71,2.71,0,0,1,10,17.51a1
.05,1.05,0,0,1,.36-.8,1.28,1.28,0,0,1,.9-
.33,1.14,1.14,0,0,1,.75.26,1.89,1.89,0,0,1,.51.73,4.54,4.54,0,0,0,.49.86,2,2
,0,0,0,.72.55A3.08,3.08,0,0,0,15,19a3,3,0,0,0,1.73-.45,1.23,1.23,0,0,0,.63-
1.06,1,1,0,0,0-.33-.8,2.3,2.3,0,0,0-.92-.49c-.39-.12-.92-.26-1.57-
.39a12.2,12.2,0,0,1-2.25-.67,3.64,3.64,0,0,1-1.48-1.06,2.47,2.47,0,0,1-.29-
.47,2.84,2.84,0,0,1-.26-1.22,2.71,2.71,0,0,1,.58-1.72,3.71,3.71,0,0,1,1.67-
1.14A7.37,7.37,0,0,1,15,9.14a7,7,0,0,1,2,.26,4.38,4.38,0,0,1,1.42.7,3.1,3.1,
0,0,1,.84.92,2.11,2.11,0,0,1,.26,1,1.13,1.13,0,0,1-.35.82,1.18,1.18,0,0,1-
.88.36,1.09,1.09,0,0,1-.74-.23,2.68,2.68,0,0,1-.51-.67,3,3,0,0,0-.77-
.94A2.42,2.42,0,0,0,14.87,11a2.76,2.76,0,0,0-1.49.36,1,1,0,0,0-
.53.81.78.78,0,0,0,.17.5,1.63,1.63,0,0,0,.52.38,4.48,4.48,0,0,0,.69.27l.15,0
,1,.24c.69.15,1.33.32,1.89.49a6.29,6.29,0,0,1,1.47.67,2.81,2.81,0,0,1,1,1,3,
3,0,0,1,.35,1.49,3.15,3.15,0,0,1-.6,1.89Zm4-2.23a7.73,7.73,0,0,0,.2-1.81c0-
.22,0-.43,0-.64a8.53,8.53,0,0,0-8.56-7.83,8.72,8.72,0,0,0-
1.46.12A5.08,5.08,0,0,0,11,6a5,5,0,0,0-
5,4.91,4.83,4.83,0,0,0,.68,2.48,7.33,7.33,0,0,0-
.13.94,6.51,6.51,0,0,0,0,.77,8.52,8.52,0,0,0,8.58,8.46,8.9,8.9,0,0,0,1.57-
.14A5.09,5.09,0,0,0,19,24a4.94,4.94,0,0,0,5-4.91,4.86,4.86,0,0,0-.52-
2.18Z'/%3e%3c/svg%3e");
}
.twitter {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.firefox {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.svg_icons {
    width: 32px;
    height: 32px;
    display: inline-block;

```

```
}
</Style>
```

HOMECONTROLLER.CS

```
public ActionResult Badge()
{
    return View();
}
```



Dot

Dot can be applied by adding the modifier class `e-badge-dot` to the target element. Dot badges are similar to notification badges, but in a minimalistic way. While using the dot badge, set the parent element to `position: relative`.

CSHTML

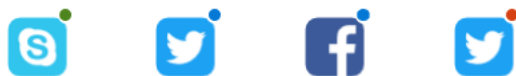
```
<div id='element'>
    <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
        <div class="skype svg_icons"></div>
        <span class="e-badge e-badge-success e-badge-overlap e-badge-dot"></span>
    </div>
    <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
        <div class="twitter svg_icons"></div>
        <span class="e-badge e-badge-info e-badge-overlap e-badge-dot"></span>
    </div>
    <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
        <div class="facebook svg_icons"></div>
        <span class="e-badge e-badge-info e-badge-overlap e-badge-dot"></span>
    </div>
    <div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
        <div class="firefox svg_icons"></div>
        <span class="e-badge e-badge-danger e-badge-overlap e-badge-dot"></span>
    </div>
</div>
<style>
#element {
    display: flex;
    width: 400px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
```

```
justify-content: center;
position: relative;
top: 130px;
}
.facebook {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%233c5a99;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M22.53,8.7h2.58V4.64H21.24a5.25,5.25,0,0,0-4,2s-1.06,1-
1.08,3.92h0v3H12.36v4.31h3.82V29h4.41V17.86h3.81.53-4.31H20.59v-
3h0A1.78,1.78,0,0,1,22.53,8.7Z'/%3e%3c/svg%3e");
}
.skype {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%2331c4ed;%7d.cls-
2%7bfill:%23fff;fill-
rule:evenodd;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2' d='M19.5,19.14a4,4,0,0,1-
1.75,1.31,7.22,7.22,0,0,1-
2.68,4.6A6.6,6.6,0,0,1,12,20.27,4,4,0,0,1,10.58,19,2.71,2.71,0,0,1,10,17.51a1
.05,1.05,0,0,1,.36-.8,1.28,1.28,0,0,1,.9-
.33,1.14,1.14,0,0,1,.75,2.6,1.89,1.89,0,0,1,.51,7.3,4.54,4.54,0,0,0,.49,8.6,2,2
,0,0,0,.72,5.5A3.08,3.08,0,0,0,15,19a3,3,0,0,0,1.73-.45,1.23,1.23,0,0,0,.63-
1.06,1,1,0,0,0-.33-.8,2.3,2.3,0,0,0-.92-.49c-.39-.12-.92-.26-1.57-
.39a12.2,12.2,0,0,1-2.25-.67,3.64,3.64,0,0,1-1.48-1.06,2.47,2.47,0,0,1-.29-
.47,2.84,2.84,0,0,1-.26-1.22,2.71,2.71,0,0,1,.58-1.72,3.71,3.71,0,0,1,1.67-
1.14A7.37,7.37,0,0,1,15,9.14a7,7,0,0,1,2,.26,4.38,4.38,0,0,1,1.42,7,3.1,3.1,
0,0,1,.84,9.2,2.11,2.11,0,0,1,.26,1,1.13,1.13,0,0,1-.35,8.2,1.18,1.18,0,0,1-
.88,3.6,1.09,1.09,0,0,1-.74-.23,2.68,2.68,0,0,1-.51-.67,3,3,0,0,0-.77-
.94A2.42,2.42,0,0,0,14.87,11a2.76,2.76,0,0,0-1.49,3.6,1,1,0,0,0-
.53,8.1,7.8,7.8,0,0,0,.17,5,1.63,1.63,0,0,0,.52,3.8,4.48,4.48,0,0,0,.69,2.7,1.15,0
,1,.24c.69,1.33,3.2,1.89,4.9a6.29,6.29,0,0,1,1.47,6.7,2.81,2.81,0,0,1,1,1,3,
3,0,0,1,.35,1.49,3.15,3.15,0,0,1-.6,1.89Zm4-2.23a7.73,7.73,0,0,0,.2-1.81c0-
.22,0-.43,0-.64a8.53,8.53,0,0,0-8.56-7.83,8.72,8.72,0,0,0-
1.46,1.2A5.08,5.08,0,0,0,11,6a5,5,0,0,0-
5,4.91,4.83,4.83,0,0,0,.68,2.48,7.33,7.33,0,0,0-
.13,9.4,6.51,6.51,0,0,0,0,.77,8.52,8.52,0,0,0,8.58,8.46,8.9,8.9,0,0,0,1.57-
.14A5.09,5.09,0,0,0,19,24a4.94,4.94,0,0,0,5-4.91,4.86,4.86,0,0,0-.52-
2.18Z'/%3e%3c/svg%3e");
}
.twitter {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231dalf2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29,6.3,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
```

```
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.firefox {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.svg_icons {
    width: 32px;
    height: 32px;
    display: inline-block;
}
.svg_icons + .e-badge.e-badge-overlap {
    transform: translateX(-100%);
}
</style>
```

HOMECONTROLLER.CS

```
public ActionResult Badge()
{
    return View();
}
```



Overlap

The overlap badge can be used with notification or dot badge, which overlaps with the target element by adding the modifier class `e-badge-overlap`. While using the overlap badge, set the parent element to `position: relative`.

CSHTML

```
<div id='element'>
    <div style="position:relative;display:inline-block;margin:20px 20px 10px
20px;">
        <div class="skype svg_icons"></div>
        <span class="e-badge e-badge-success e-badge-overlap e-badge-
notification">99+</span>
    </div>
```

```

<div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="twitter svg_icons"></div>
    <span class="e-badge e-badge-info e-badge-overlap e-badge-notification">27</span>
</div>
<div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="facebook svg_icons"></div>
    <span class="e-badge e-badge-info e-badge-overlap e-badge-notification">2</span>
</div>
<div style="position:relative;display:inline-block;margin:20px 20px 10px 20px;">
    <div class="firefox svg_icons"></div>
    <span class="e-badge e-badge-danger e-badge-overlap e-badge-notification">35</span>
</div>
</div>
<Style>
#element {
    display: flex;
    width: 400px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
    justify-content: center;
    position: relative;
    top: 130px;
}
.facebook {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%233c5a99;%7d.cls-2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect class='cls-1' x='1' y='1' width='28' height='28' rx='5.75' ry='5.75'/%3e%3cpath class='cls-2' d='M22.53,8.7h2.58V4.64H21.24a5.25,5.25,0,0,0-4,2s-1.06,1-1.08,3.92h0v3H12.36v4.31h3.82V29h4.41V17.86h3.81.53-4.31H20.59v-3h0A1.78,1.78,0,0,1,22.53,8.7Z'/%3e%3c/svg%3e");
}
.skype {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%2331c4ed;%7d.cls-2%7bfill:%23fff;fill-rule:evenodd;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect class='cls-1' x='1' y='1' width='28' height='28' rx='5.75' ry='5.75'/%3e%3cpath class='cls-2' d='M19.5,19.14a4,4,0,0,1-1.75,1.31,7.22,7.22,0,0,1-2.68,4.6A6.6,6.6,0,0,1,12,20.27,4,4,0,0,1,10.58,19,2.71,2.71,0,0,1,10,17.51a1.05,1.05,0,0,1,.36-.8,1.28,1.28,0,0,1,.9-.33,1.14,1.14,0,0,1,.75,2.6,1.89,1.89,0,0,1,.51,7.3,4.54,4.54,0,0,0,.49,8.6,2,2,0,0,0,.72,5.5A3.08,3.08,0,0,0,15,19a3,3,0,0,0,1.73-.45,1.23,1.23,0,0,0,.63-1.06,1,1,0,0,0-.33-.8,2.3,2.3,0,0,0-.92-.49c-.39-.12-.92-.26-1.57-.39a12.2,12.2,0,0,1-2.25-.67,3.64,3.64,0,0,1-1.48-1.06,2.47,2.47,0,0,1-.29-.47,2.84,2.84,0,0,1-.26-1.22,2.71,2.71,0,0,1,.58-1.72,3.71,3.71,0,0,1,1.67-

```

```

1.14A7.37,7.37,0,0,1,15,9.14a7,7,0,0,1,2,.26,4.38,4.38,0,0,1,1.42.7,3.1,3.1,
0,0,1,.84.92,2.11,2.11,0,0,1,.26,1,1.13,1.13,0,0,1-.35.82,1.18,1.18,0,0,1-
.88.36,1.09,1.09,0,0,1-.74-.23,2.68,2.68,0,0,1-.51-.67,3,3,0,0,0-.77-
.94A2.42,2.42,0,0,0,14.87,11a2.76,2.76,0,0,0-1.49.36,1,1,0,0,0-
.53.81.78.78,0,0,0,.17.5,1.63,1.63,0,0,0,.52.38,4.48,4.48,0,0,0,.69.271.15,0
,1,.24c.69.15,1.33.32,1.89.49a6.29,6.29,0,0,1,1.47.67,2.81,2.81,0,0,1,1,1,3,
3,0,0,1,.35,1.49,3.15,3.15,0,0,1-.6,1.89Zm4-2.23a7.73,7.73,0,0,0,.2-1.81c0-
.22,0-.43,0-.64a8.53,8.53,0,0,0-8.56-7.83,8.72,8.72,0,0,0-
1.46.12A5.08,5.08,0,0,0,11,6a5,5,0,0,0-
5,4.91,4.83,4.83,0,0,0,.68,2.48,7.33,7.33,0,0,0-
.13.94,6.51,6.51,0,0,0,0,.77,8.52,8.52,0,0,0,8.58,8.46,8.9,8.9,0,0,0,1.57-
.14A5.09,5.09,0,0,0,19,24a4.94,4.94,0,0,0,5-4.91,4.86,4.86,0,0,0-.52-
2.18Z'/%3e%3c/svg%3e");
}
.twitter {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.firefox {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}
.svg_icons {
    width: 32px;
    height: 32px;
    display: inline-block;
}
</Style>

```

HOMECONTROLLER.CS

```

public ActionResult Badge()
{
    return View();
}

```

```
}

```



Position

The default position of the notification or dot badge is top. But, the position can be changed to bottom using the modifier class `.e-badge-bottom`. For example, the bottom class modifier is used with dot badge to display the status in the avatar as shown in the following sample.

CSHTML

```
<div id='element'>
  <div class="badge-block">
    <div class="contact svg_icons"></div>
    <!-- Success Colored Bottom Dot Badge -->
    <span class="e-badge e-badge-success e-badge-overlap e-badge-dot e-
badge-bottom"></span>
  </div>
  <div class="badge-block">
    <div class="skype svg_icons"></div>
    <!-- Info Colored Bottom Dot Badge -->
    <span class="e-badge e-badge-info e-badge-overlap e-badge-dot e-
badge-bottom"></span>
  </div>
  <div class="badge-block">
    <div class="facebook svg_icons"></div>
    <!-- Info Colored Dot Badge -->
    <span class="e-badge e-badge-info e-badge-overlap e-badge-
dot"></span>
  </div>
  <div class="badge-block">
    <div class="pinterest svg_icons"></div>
    <!-- Danger Colored Dot Badge -->
    <span class="e-badge e-badge-danger e-badge-overlap e-badge-
dot"></span>
  </div>
</div>
<style>
#element {
  display: flex;
  width: 400px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
  justify-content: center;
  position: relative;
  top: 130px;
}
.badge-block {
  position: relative;
  display: inline-block;
  margin: 10px 13px 10px 10px;
}
```



```

/* SVG Icons */
.facebook {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%233c5a99;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M22.53,8.7h2.58V4.64H21.24a5.25,5.25,0,0,0-4.2s-1.06,1-
1.08,3.92h0v3H12.36v4.31h3.82V29h4.41V17.86h3.81.53-4.31H20.59v-
3h0A1.78,1.78,0,0,1,22.53,8.7Z'/%3e%3c/svg%3e");
}

.skype {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%2331c4ed;%7d.cls-
2%7bfill:%23fff;fill-
rule:evenodd;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2' d='M19.5,19.14a4,4,0,0,1-
1.75,1.31,7.22,7.22,0,0,1-
2.68,4.6A6.6,6.6,0,0,1,12,20.27,4,4,0,0,1,10.58,19,2.71,2.71,0,0,1,10,17.51a1
.05,1.05,0,0,1,.36-.8,1.28,1.28,0,0,1,.9-
.33,1.14,1.14,0,0,1,.75,2.6,1.89,1.89,0,0,1,.51,7.3,4.54,4.54,0,0,0,.49,8.6,2,2
,0,0,0,.72,5.5A3.08,3.08,0,0,0,15,19a3,3,0,0,0,1.73-.45,1.23,1.23,0,0,0,.63-
1.06,1,1,0,0,0-.33-.8,2.3,2.3,0,0,0-.92-.49c-.39-.12-.92-.26-1.57-
.39a12.2,12.2,0,0,1-2.25-.67,3.64,3.64,0,0,1-1.48-1.06,2.47,2.47,0,0,1-.29-
.47,2.84,2.84,0,0,1-.26-1.22,2.71,2.71,0,0,1,.58-1.72,3.71,3.71,0,0,1,1.67-
1.14A7.37,7.37,0,0,1,15,9.14a7,7,0,0,1,2,.26,4.38,4.38,0,0,1,1.42,7,3.1,3.1,
0,0,1,.84,9.2,2.11,2.11,0,0,1,.26,1,1.13,1.13,0,0,1-.35,8.2,1.18,1.18,0,0,1-
.88,3.6,1.09,1.09,0,0,1-.74-.23,2.68,2.68,0,0,1-.51-.67,3,3,0,0,0-.77-
.94A2.42,2.42,0,0,0,14.87,11a2.76,2.76,0,0,0-1.49,3.6,1,1,0,0,0-
.53,8.1,7.8,7.8,0,0,0,.17,5,1.63,1.63,0,0,0,.52,3.8,4.48,4.48,0,0,0,.69,2.7,1.15,0
,1,.24c.69,1.5,1.33,3.2,1.89,4.9a6.29,6.29,0,0,1,1.47,6.7,2.81,2.81,0,0,1,1,1,3,
3,0,0,1,.35,1.49,3.15,3.15,0,0,1-.6,1.89Zm4-2.23a7.73,7.73,0,0,0,.2-1.81c0-
.22,0-.43,0-.64a8.53,8.53,0,0,0-8.56-7.83,8.72,8.72,0,0,0-
1.46,1.2A5.08,5.08,0,0,0,11,6a5,5,0,0,0-
5,4.91,4.83,4.83,0,0,0,.68,2.48,7.33,7.33,0,0,0-
.13,9.4,6.51,6.51,0,0,0,0,.77,8.52,8.52,0,0,0,8.58,8.46,8.9,8.9,0,0,0,1.57-
.14A5.09,5.09,0,0,0,19,24a4.94,4.94,0,0,0,5-4.91,4.86,4.86,0,0,0-.52-
2.18Z'/%3e%3c/svg%3e");
}

.twitter {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3e%3cdefs%3e%3cstyle%3e.cls-1%7bfill:%231da1f2;%7d.cls-
2%7bfill:%23fff;%7d%3c/style%3e%3c/defs%3e%3ctitle%3eIcon%3c/title%3e%3crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3e%3cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29,6.3,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3e%3c/svg%3e");
}

```

```

.whatsapp {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#25D366' d='M11.863 8.305c-.444-.011-.813.335-1.18.677-.12.11-
.235.228-.344.35-.489.552-.798 1.186-.703 1.935.103.807.187 1.63.423
2.402.65 2.13 1.962 3.843 3.605 5.309 1.51 1.346 3.162 2.47 5.151
2.977a9.847 9.847 0 0 0 1.966.268c.984.044 2.378-.942 2.728-1.86a.37.37 0 0
0 .022-.158c-.028-.435-.052-.87-.097-1.303-.009-.088-.083-.212-.158-
.242a276.61 276.61 0 0 0-3.316-1.281c-.277-.105-.526-.057-.749.159-.337.325-
.707.617-1.029.956-.207.217-.384.257-.617.078-.619-.48-1.26-.933-1.853-
1.443-.93-.8-1.7-1.744-2.38-2.763-.19-.284-.199-.483.077-.724.294-.256.517-
.592.804-.856.366-.336.37-.69.192-1.12a85.3 85.3 0 0 1-.964-2.46c-.3-.792-
.294-.795-1.258-.833a.894.894 0 0 0-.32-.068zm4.016-5.566c5.806-.062 10.95
3.973 12.154 9.89 1.123 5.523-1.825 11.269-6.975 13.536-3.56 1.568-7.1
1.477-10.606-.216-.13-.063-.337-.05-.479.004-1.39.532-2.777 1.08-4.164
1.625-.069.027-.14.049-.266.093.412-1.455.8-2.847 1.205-4.235.064-.216.03-
.357-.112-.526-2.056-2.454-3.018-5.292-2.889-8.483.226-5.636 4.415-10.442
9.968-11.475.727-.136 1.45-.205 2.164-.213zm.203-2.086a14.416 14.416 0 0 0-
4.595.74c-5.636 1.87-9.435 6.872-9.764 12.803-.189 3.43.773 6.54 2.82
9.307.133.182.178.333.12.563-.576 2.266-1.137 4.535-1.718 6.866.178-.065.31-
.111.438-.162 2.147-.838 4.297-1.67 6.44-2.521.324-.128.598-.141.925-.005
1.091.452 2.228.737 3.4.883 3.548.44 6.831-.306 9.797-2.291 5.426-3.631
7.685-10.342 5.562-16.54A14.243 14.243 0 0 0 16.082.654zM16.032
0c.494.004.992.03 1.492.078 6.504.613 12.026 5.686 13.158 12.108.805 4.565-
.231 8.687-3.16 12.275-2.333 2.857-5.366 4.599-9.012 5.227a14.618 14.618 0 0
1-7.895-.793.818.818 0 0 0-.649 0c-2.55 1.005-5.105 1.999-7.659 2.996-
.08.031-.162.058-.305.1081.783-3.13c.393-1.573.791-3.144 1.172-4.72a.617.617
0 0 0-.089-.442c-1.92-2.71-2.94-5.725-2.8-9.04.26-6.171 3.233-10.635 8.781-
13.33C11.81.386 13.89-.017 16.032 0z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}

.firefox {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3Csvg
id='Layer_1' data-name='Layer 1' xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 32 32'%3E%3Cdefs%3E%3Cstyle%3E.cls-1%7Bfill:%231da1f2;%7d.cls-
2%7Bfill:%23fff;%7d%3C/style%3E%3C/defs%3E%3Ctitle%3EIcon%3C/title%3E%3Crect
class='cls-1' x='1' y='1' width='28' height='28' rx='5.75'
ry='5.75'/%3E%3Cpath class='cls-2'
d='M11.67,23.11A11.3,11.3,0,0,0,23.05,11.73c0-.17,0-.35,0-.52a8,8,0,0,0,2-
2.07,8,8,0,0,1-2.29.63,4,4,0,0,0,1.75-2.21,8,8,0,0,1-2.54,1,4,4,0,0,0-
6.81,3.65A11.36,11.36,0,0,1,6.89,8a4,4,0,0,0-
.54,2,4,4,0,0,0,1.78,3.33,4,4,0,0,1-1.81-
.5v.05a4,4,0,0,0,3.2,3.92,4,4,0,0,1-1,.14,3.67,3.67,0,0,1-.75-
.07,4,4,0,0,0,3.74,2.78,8.06,8.06,0,0,1-5,1.71,7.61,7.61,0,0,1-1-
.06,11.31,11.31,0,0,0,6.14,1.8'/%3E%3C/svg%3E");
}

.contact {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cellipse cx='16'
cy='16' fill='#4285f4' rx='16' ry='16'/%3E%3Cpath fill='#FFF' d='M13.55
16.95h4.9c2.7 0 4.85 2.05 4.85 4.6 0 .9-.25 1.75-.75 2.45H9.45c-.5-.7-.75-
1.55-.75-2.45 0-2.55 2.15-4.6 4.85-4.6zM16.05 8c2.05 0 3.7 1.65 3.7 3.7 0
2.05-1.65 3.7-3.7 3.7-2.05 0-3.7-1.65-3.7-3.7.05-2.05 1.7-3.7 3.7-
3.7z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}

.chrome {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath

```

```

fill='#ffffff' d='M16.033 11.049a5.155 5.155 0 1 1 0 10.312 5.156 5.156 0 0
1 0-10.312zM16.124 0c1.281-.003 9.659.318 14.268 9.043h-.016l.01.018c.33.578
3.745 6.94-.485 14.969 0 0-4.215 8.107-14.565 7.968l-.452-.012-.004.007-
.004.007.02-.037c.564-.98 5.112-8.884 6.357-11.067l.016-.028.007-.008.04-
.069.11-.127a7.085 7.085 0 0 0 1.457-2.967l.01-.046.035-.151c.088-.424.148-
.944.128-1.549l-.006-.117v-.004l-.007-.143-.006-.07-.005-.079-.012-.116v-
.011-.001-.008-.016-.158a7.2 7.2 0 0 0-.096-.572l-.018-.081-.013-.064a9.801
9.801 0 0 0-.692-2.016c-.165-.243-.332-.489-.512-.733l-.142-.187 8.728-
2.554s-10.538-.01-13.018-.001l.021.005H16.642l-.14-.013a7.034 7.034 0 0 0-
1.132-.003l-.167.016h-.047l-.034-.001c-.193.002-1.213.045-2.492.764l-
.005.003-.033.016a7.158 7.158 0 0 0-3.25 3.533l-.059.148-6.485-6.404s4.74
8.311 6.165 10.779l.065.113.023.088a7.14 7.14 0 0 0 7.777 5.118l.144-
.02L14.854 32h-.027c-.667-.005-7.894-.234-12.744-7.906 0 0-4.925-7.698.37-
16.573l.252-.411.001-.002C2.822 6.904 6.58.374 15.958.003c0 0 .057-.003.166-
.003z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.pinterest {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cellipse cx='16'
cy='16' fill='#bd081c' rx='16' ry='16'/%3E%3Cpath fill='#FFF' d='M16.22
6.458c4.807-.009 9.028 1.888 9.663 5.307.787 4.256-2.438 8.866-8.213 8.535-
1.565-.09-2.222-.666-3.448-1.22-.675 2.628-1.5 5.147-3.942 6.463-.754-3.972
1.107-6.954 1.971-10.12-1.474-1.842.177-5.547 3.284-4.634 3.824 1.123-3.31
6.845 1.48 7.56 5 .746 7.04-6.441 3.94-8.779-4.48-3.376-13.042-.077-11.989
4.755.256 1.181 1.9 1.54.657 3.17-2.868-.471-3.724-2.15-3.614-4.39.178-3.664
4.435-6.229 8.705-6.583.506-.043 1.01-.064 1.507-.064z'/%3E%3C/svg%3E") no-
repeat 100% 100%;
}
.svg_icons {
    width: 32px;
    height: 32px;
    display: inline-block;
}
.svg_icons + .e-badge.e-badge-overlap {
    transform: translateX(-100%);
}
</style>

```

HOMECONTROLLER.CS

```

public ActionResult Badge()
{
    return View();
}

```



Note: [View Sample in GitHub.](#)

How To

Customization in ASP.NET MVC Badge Control

Colour customization


Even though badges come with 8 predefined colors, you can also customize the colour of the badge as desired.

CSHTML

```
<div id='element'>
    <h1>Color Customization <span class="e-badge e-badge-primary e-badge-
pill green">New</span></h1>
    <h1>Color Customization <span class="e-badge e-badge-primary e-badge-
pill bue">New</span></h1>
    <h1>Color Customization <span class="e-badge e-badge-primary e-badge-
pill purple">New</span></h1>
    <h1>Color Customization <span class="e-badge e-badge-primary e-badge-
pill gradient">New</span></h1>
</div>
<style>
#element {
    display: table;
    width: 560px;
    margin: auto;
    height: 200px;
    border: 1px solid #dddddd;
    border-radius: 3px;
    justify-content: center;
    position: relative;
    top: 55px;
}
h1 {
    text-align: center;
}
#element .e-badge.green {
    background: #329378;
    color: #fff;
}
#element .e-badge.blue {
    background: #5f65b8;
    color: #fff;
}
#element .e-badge.gradient {
    background: #4776E6;
    /* fallback for old browsers */
    background: -webkit-linear-gradient(to top, #8E54E9, #4776E6);
    /* Chrome 10-25, Safari 5.1-6 */
    background: linear-gradient(to top, #8E54E9, #4776E6);
    /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
    color: #fff;
}
#element .e-badge.purple {
    background: #9a428f;
    color: #fff;
}
</style>
```

HOMECONTROLLER.CS

```
public ActionResult Badge()  
{  
    return View();  
}
```

Color Customization 

Color Customization 

Color Customization 

Color Customization 

Customize badge size

Badges are designed to change its size based on the content. To change the size of a badge, adjust the font size of the badge.

CSHTML

```
<div id='element'>  
    <h1>Badge Component <span class="e-badge e-badge-primary  
size_1">New</span></h1>  
    <h1>Badge Component <span class="e-badge e-badge-primary  
size_2">New</span></h1>  
    <h1>Badge Component <span class="e-badge e-badge-primary  
size_3">New</span></h1>  
</div>  
<Style>  
#element {  
    display: block;  
    width: 400px;  
    margin: auto;  
    border: 1px solid #dddddd;  
    border-radius: 3px;  
    justify-content: center;  
}  
h1 {  
    text-align: center;  
}  
.e-badge.size_1 {  
    font-size: 12px;  
}  
.e-badge.size_2 {
```

```
font-size: 16px;
}
.e-badge.size_3 {
    font-size: 18px;
}
</Style>
```

HOMECONTROLLER.CS

```
public ActionResult Badge()
{
    return View();
}
```

Badge Component New

Badge Component New

Badge Component New

Custom position

Even though the badges support the conventional **top** and **bottom** positions, the position of the badges can be changed as desired. This can be done by adding a custom class to the badge element to override the default position applied from the source.

CSHTML

```
<div id='element'>
    <div style="width: 200px;margin: 10px auto;">
        <div class="badge-block">
            <div class="whatsapp svg_icons"></div>
            <!-- Warning Colored Notification Badge -->
            <span class="e-badge e-badge-warning e-badge-notification e-
badge-overlap leftTop">99+</span>
        </div>
        <div class="badge-block">
            <div class="facebook svg_icons"></div>
            <!-- Danger Colored Notification Badge -->
            <span class="e-badge e-badge-danger e-badge-notification e-
badge-overlap leftTop">99+</span>
        </div>
        <div class="badge-block">
            <div class="skype svg_icons"></div>
            <!-- Secondary Colored Notification Badge -->
            <span class="e-badge e-badge-secondary e-badge-notification e-
badge-overlap leftTop">18</span>
        </div>
    </div>
```

```

    <div style="width: 200px;margin: 10px auto;">
      <div class="badge-block">
        <div class="whatsapp svg_icons"></div>
        <!-- Warning Colored Notification Badge -->
        <span class="e-badge e-badge-warning e-badge-notification e-
badge-overlap leftBottom">99+</span>
      </div>
      <div class="badge-block">
        <div class="facebook svg_icons"></div>
        <!-- Danger Colored Notification Badge -->
        <span class="e-badge e-badge-danger e-badge-notification e-
badge-overlap leftBottom">99+</span>
      </div>
      <div class="badge-block">
        <div class="skype svg_icons"></div>
        <!-- Secondary Colored Notification Badge -->
        <span class="e-badge e-badge-secondary e-badge-notification e-
badge-overlap leftBottom">18</span>
      </div>
    </div>
</div>
<style>
#element {
  display: flex;
  width: 500px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
  justify-content: center;
  position: relative;
  top: 130px;
}
.badge-block {
  position: relative;
  display: inline-block;
  margin: 10px 13px 10px 10px;
}
.badge-block .e-badge.leftBottom {
  transform: translateX(-150%) translateY(200%);
}
.badge-block .e-badge.leftTop {
  transform: translateX(-150%);
}
/* SVG Icons */
.facebook {
  background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#3b5998' d='M1.82 0h28.36c.908 0 1.645.819 1.645 1.829v28.342c0 1.011-
.737 1.829-1.645 1.829h-8.188V18.852h3.934v-4.271h-3.934V13.15c0-1.04.828-
1.883 1.853-1.883h2.081V6.998h-4.505c-2.737 0-4.955 2.251-4.955
5.029v2.554h-3.55v4.271h3.55V32H1.82c-.908 0-1.645-.818-1.645-
1.829V1.829C.175.819.912 0 1.82 0z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.skype {
  background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#00aff0' d='M16.067 5.579c-1.68 0-3.166.234-4.416.696-1.265.468-2.247

```

```
1.15-2.92 2.028a4.893 4.893 0 0 0-1.022 3.046c0 1.192.328 2.207.974
3.016.636.796 1.505 1.433 2.585 1.891 1.056.449 2.383.847 3.944 1.181
1.148.24 2.077.471 2.76.684.657.204 1.198.502 1.606.885.392.367.582.833.582
1.428 0 .752-.364 1.366-1.114 1.878-.767.523-1.787.789-3.031.789-.905 0-
1.64-.131-2.187-.389-.541-.257-.965-.585-1.261-.976-.308-.406-.599-.922-
.864-1.533-.24-.56-.537-.995-.883-1.288-.363-.307-.809-.462-1.326-.462-.63
0-1.157.196-1.57.584a1.89 1.89 0 0 0-.628 1.421c0 .882.325 1.797.964
2.719a7.062 7.062 0 0 0 2.478 2.199c1.413.75 3.226 1.131 5.388 1.131 1.8 0
3.383-.278 4.702-.827 1.333-.554 2.362-1.335 3.058-2.32a5.711 5.711 0 0 0
1.053-3.358c0-1.037-.206-1.928-.612-2.65a5.125 5.125 0 0 0-1.697-1.792c-
.706-.46-1.572-.856-2.574-1.176-.99-.317-2.11-.61-3.327-.871a55.674 55.674 0
0 1-2.082-.51 6.642 6.642 0 0 1-1.211-.47c-.382-.191-.683-.42-.897-
.681a1.328 1.328 0 0 1-.301-.876c0-.559.306-1.031.933-1.443.652-.427 1.529-
.643 2.608-.643 1.16 0 2.006.195 2.512.579.52.397.976.959 1.35
1.672.325.558.617.946.898 1.195.302.268.74.404 1.297.404.613 0 1.133-.217
1.545-.646.41-.425.617-.914.617-1.452 0-.559-.158-1.136-.47-1.717-.31-.574-
.801-1.127-1.463-1.644-.657-.514-1.493-.931-2.485-1.24-.987-.306-2.168-.462-
3.513-.462zM8.95 0c1.708 0 3.298.492 4.643 1.339.836-.144 1.696-.22 2.575-
.22 8.31 0 15.049 6.738 15.049 15.049 0 1.109-.121 2.189-.35 3.229a8.728
8.728 0 0 1-11.945 11.567c-.892.166-1.814.253-2.754.253-8.311 0-15.05-6.738-
15.05-15.049 0-1.037.105-2.049.304-3.026A8.727 8.727 0 0 1 8.95
0z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.twitter {
  background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#1da1f2' d='M22.155 5.26c1.888 0 3.594.658 4.792 1.712a14.9 14.9 0 0 0
4.169-1.316c-.49 1.267-1.531 2.33-2.887 3A15.3 15.3 0 0 0 32 7.802a12.298
12.298 0 0 1-3.276 2.807c.013.233.019.467.019.703 0 7.164-6.604 15.427-
18.679 15.427-3.708 0-7.158-.897-10.064-2.436.514.05 1.037.076 1.566.076
3.076 0 5.907-.867 8.153-2.322-2.872-.043-5.297-1.612-6.132-3.766a7.864
7.864 0 0 0 2.964-.093c-3.003-.498-5.266-2.688-5.266-5.316l.001-.067a7.637
7.637 0 0 0 2.974.678c-1.762-.974-2.921-2.633-2.921-4.514 0-.994.324-
1.925.889-2.726 3.238 3.28 8.075 5.438 13.532 5.666a4.542 4.542 0 0 1-.17-
1.237c0-2.994 2.939-5.421 6.565-5.421z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.whatsapp {
  background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#25D366' d='M11.863 8.305c-.444-.011-.813.335-1.18.677-.12.11-
.235.228-.344.35-.489.552-.798 1.186-.703 1.935.103.807.187 1.63.423
2.402.65 2.13 1.962 3.843 3.605 5.309 1.51 1.346 3.162 2.47 5.151
2.977a9.847 9.847 0 0 0 1.966.268c.984.044 2.378-.942 2.728-1.86a37.37 0 0
0 .022-.158c-.028-.435-.052-.87-.097-1.303-.009-.088-.083-.212-.158-
.242a276.61 276.61 0 0 0-3.316-1.281c-.277-.105-.526-.057-.749.159-.337.325-
.707.617-1.029.956-.207.217-.384.257-.617.078-.619-.48-1.26-.933-1.853-
1.443-.93-.8-1.7-1.744-2.38-2.763-.19-.284-.199-.483.077-.724.294-.256.517-
.592.804-.856.366-.336.37-.69.192-1.12a85.3 85.3 0 0 1-.964-2.46c-.3-.792-
.294-.795-1.258-.833a.894.894 0 0 0-.32-.068zm4.016-5.566c5.806-.062 10.95
3.973 12.154 9.89 1.123 5.523-1.825 11.269-6.975 13.536-3.56 1.568-7.1
1.477-10.606-.216-.13-.063-.337-.05-.479.004-1.39.532-2.777 1.08-4.164
1.625-.069.027-.14.049-.266.093.412-1.455.8-2.847 1.205-4.235.064-.216.03-
.357-.112-.526-2.056-2.454-3.018-5.292-2.889-8.483.226-5.636 4.415-10.442
9.968-11.475.727-.136 1.45-.205 2.164-.213zm.203-2.086a14.416 14.416 0 0 0-
4.595.74c-5.636 1.87-9.435 6.872-9.764 12.803-.189 3.43.773 6.54 2.82
9.307.133.182.178.333.12.563-.576 2.266-1.137 4.535-1.718 6.866.178-.065.31-
.111.438-.162 2.147-.838 4.297-1.67 6.44-2.521.324-.128.598-.141.925-.005
```



```
1.091.452 2.228.737 3.4.883 3.548.44 6.831-.306 9.797-2.291 5.426-3.631
7.685-10.342 5.562-16.54A14.243 14.243 0 0 0 16.082.654zM16.032
0c.494.004.992.03 1.492.078 6.504.613 12.026 5.686 13.158 12.108.805 4.565-
.231 8.687-3.16 12.275-2.333 2.857-5.366 4.599-9.012 5.227a14.618 14.618 0 0
1-7.895-.793.818.818 0 0 0-.649 0c-2.55 1.005-5.105 1.999-7.659 2.996-
.08.031-.162.058-.305.1081.783-3.13c.393-1.573.791-3.144 1.172-4.72a.617.617
0 0 0-.089-.442c-1.92-2.71-2.94-5.725-2.8-9.04.26-6.171 3.233-10.635 8.781-
13.33C11.81.386 13.89-.017 16.032 0z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.firefox {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#e66000' d='M27.579 8.9021.048.09a.54.54 0 0 1-.045-.082zm-7.06-
4.054c-.351.003-.712.024-1.081.067 1.2.192 2.284.501 2.729 1.457-.625-.005-
1.3-.054-1.58.291 2.275 1.189 5.035 1.886 5.315 5.096-.36-.17-.48-.58-1.005-
.582.23 1.595.78 3.799.145 5.388-.35-.422-.365-1.185-.865-1.457.105 2.68-
.025 5.119-1.29 6.407-.035-.572.405-1.424 0-1.892-1.15 3.86-8.29 5.12-10.2
1.746 3.016.292 3.73-1.751 6.176-2.038-.125-.937-.915-1.676-2.01-1.748-1.19-
.076-2.215 1.096-3.305 1.02-.56-.04-1.18-.554-1.725-1.02-1.77-1.514-.894-
2.931 1.58-1.892.325-.797-.145-1.698-.43-2.33.64-.954 2.245-.928 2.3-2.475-
1.76-.012-2.745-.81-3.45-1.894.33-1.317 1.165-2.124 2.015-2.912-1.65.172-
2.605 1.05-3.734 1.748-1.075-.346-2.275.005-3.16.29-1-.539-1.15-1.941-1.58-
3.057-1.1 1.167-1.445 3.102-1.44 5.388-.765 1.018-1.57 1.998-1.575
3.786.42.168.53-.698.715-.292-2.335 8.76 5.49 16.323 13.644 16.018 8.354-
.313 13.649-8.231 13.219-17.182-.65-.12.03 1.098-.575 1.019.025-2.797-.825-
5.474-2.155-6.407.31.28.204.983.35 1.4281.032.083-.083-.157c-1.265-2.286-
3.702-3.923-6.977-3.897zM16.103 0c2.763-.007 5.09.615 6.784 1.421 4.995
2.381 9.719 7.97 9.049 16.162-.73 8.88-8.514 14.467-16.088 14.417C6.794
31.937-.645 24.6.045 14.816.32 10.992 1.724 8.529 2.634 7.1 4.599 3.998
8.959.67 14.123.111a18.887 18.887 0 0 1 1.98-.11z'/%3E%3C/svg%3E") no-repeat
100% 100%;
}
.contact {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cellipse cx='16'
cy='16' fill='#4285f4' rx='16' ry='16'/%3E%3Cpath fill='#FFF' d='M13.55
16.95h4.9c2.7 0 4.85 2.05 4.85 4.6 0 .9-.25 1.75-.75 2.45H9.45c-.5-.7-.75-
1.55-.75-2.45 0-2.55 2.15-4.6 4.85-4.6zM16.05 8c2.05 0 3.7 1.65 3.7 3.7 0
2.05-1.65 3.7-3.7 3.7-2.05 0-3.7-1.65-3.7-3.7.05-2.05 1.7-3.7 3.7-
3.7z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.chrome {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='#ffffff' d='M16.033 11.049a5.155 5.155 0 1 1 0 10.312 5.156 5.156 0 0
1 0-10.312zM16.124 0c1.281-.003 9.659.318 14.268 9.043h-.016l.01.018c.33.578
3.745 6.94-.485 14.969 0 0-4.215 8.107-14.565 7.968l-.452-.012-.004.007-
.004.007.02-.037c.564-.98 5.112-8.884 6.357-11.067l.016-.028.007-.008.04-
.069.11-.127a7.085 7.085 0 0 0 1.457-2.967l.01-.046.035-.151c.088-.424.148-
.944.128-1.549l-.006-.117v-.004l-.007-.143-.006-.07-.005-.079-.012-.116v-
.011-.001-.008-.016-.158a7.2 7.2 0 0 0-.096-.572l-.018-.081-.013-.064a9.801
9.801 0 0 0-.692-2.016c-.165-.243-.332-.489-.512-.733l-.142-.187 8.728-
2.554s-10.538-.01-13.018-.001l.021.005H16.642l-.14-.013a7.034 7.034 0 0 0-
1.132-.003l-.167.016h-.047l-.034-.001c-.193.002-1.213.045-2.492.764l-
.005.003-.033.016a7.158 7.158 0 0 0-3.25 3.533l-.059.148-6.485-6.404s4.74
8.311 6.165 10.779l.065.113.023.088a7.14 7.14 0 0 0 7.777 5.118l.144-
.02L14.854 32h-.027c-.667-.005-7.894-.234-12.744-7.906 0 0-4.925-7.698.37-
```

```

16.5731.252-.411.001-.002C2.822 6.904 6.58.374 15.958.003c0 0 .057-.003.166-
.003z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.pinterest {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cellipse cx='16'
cy='16' fill='#bd081c' rx='16' ry='16'/%3E%3Cpath fill='#FFF' d='M16.22
6.458c4.807-.009 9.028 1.888 9.663 5.307.787 4.256-2.438 8.866-8.213 8.535-
1.565-.09-2.222-.666-3.448-1.22-.675 2.628-1.5 5.147-3.942 6.463-.754-3.972
1.107-6.954 1.971-10.12-1.474-1.842.177-5.547 3.284-4.634 3.824 1.123-3.31
6.845 1.48 7.56 5 .746 7.04-6.441 3.94-8.779-4.48-3.376-13.042-.077-11.989
4.755.256 1.181 1.9 1.54.657 3.17-2.868-.471-3.724-2.15-3.614-4.39.178-3.664
4.435-6.229 8.705-6.583.506-.043 1.01-.064 1.507-.064z'/%3E%3C/svg%3E") no-
repeat 100% 100%;
}
.svg_icons {
    width: 32px;
    height: 32px;
    display: inline-block;
}
</style>

```

HOMECONTROLLER.CS

```

public ActionResult Badge()
{
    return View();
}

```



Note: [View Sample in GitHub.](#)

Integrate Badge into ListView

The badges can be integrated with the `listview` component to indicate new notification with colour based on priority.

In the following sample, `default` badges are used and there is no need to customize the badge size. The component will automatically adjust the size based on the container element.

CSHTML

```

@using Syncfusion.EJ2
@{
    var template = "<div class='listWrapper' style='width: inherit;
height: inherit;'>" +
    "        <span class='list_svg'> </span>" +
    "        <span class='list_text'>${text}</span>" +
    "        ${if(messages != '')} +
    "        <span class='${badge}' style='float: right; margin-top: 16px;
font-size: 12px;'>${messages}</span>" +
    "        ${/if}" +

```

```

"        </div>";
}

<div class="col-lg-12 control-section">
    <div class="sample_container badge-list">
        <!-- Listview element -->

@Html.EJS().ListView("lists").Enable(true).DataSource((IEnumerable<object>)V
iewBag.dataSource).HeaderTitle("Inbox").ShowHeader(true).Template(template).
ActionComplete("actionComplete").Fields(new
Syncfusion.EJ2.Lists.ListViewFieldSettings { GroupBy = "type" }).Render()
    </div>
</div>
<script>
    function actionComplete() {
        let list =
document.getElementById('lists').getElementsByClassName('e-list-group-
item')[0];
        list.style.display = 'none';
    }
</script>
<style>
    .control-section {
        overflow: auto;
    }
    .sample_container.badge-list {
        max-width: 350px;
        margin: auto;
    }
    #lists {
        margin: auto;
        border: 1px solid rgba(0, 0, 0, 0.12)
    }
    #lists .e-list-item {
        cursor: pointer;
        height: 50px;
        line-height: 48px;
        border: 0;
    }
    /* SVG Icons and Customization */
    .list_svg {
        width: 24px;
        height: 24px;
        display: inline-block;
        margin-top: 11px;
        margin-right: 16px;
    }
    .list_text {
        width: 60%;
        display: inline-block;
        vertical-align: top;
    }
    .updates {
        background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M17.024 13.8121.022.162v.893c0 .487-.007.942-.022 1.366-
.012.423-.027.833-.042 1.23v2.435c0 .415.022.799.064
1.15.042.35.11.559.202.622.091.063.232.095.422.095h.3281.105.324c-.041.073-

```

```
.125.136-.253.189-.127.055-.226.063-.296.028H14.126a.556.556 0 0 1-.34-
.109v-.27c.086-.108.22-.162.403-.162h.38c.185-.109.294-.219.328-.328.036-
.108.054-.3.054-.573a7.32 7.32 0 0 0 .042-.819c0-.29.007-.527.021-
.709V17.233c0-.636-.018-1.041-.053-1.213-.034-.173-.082-.386-.138-.64a20.09
20.09 0 0 1-.455.19c-.176.072-.291.108-.348.1081-.105-.162c0-.162.09-
.307.274-.4341.38-.27a6.472 6.472 0 0 1 .847-.406c.227-.144.469-.265.73-
.365.26-.1.554-.176.878-.229zm-.952-5.736a.9.9 0 0 1
.613.243c.184.162.343.361.477.595.135.235.2.451.2.65 0 .451-.129.802-.39
1.054-.261.254-.453.397-.572.434a1.424 1.424 0 0 1-.412.054.934.934 0 0 1-
.455-.122c-.15-.082-.293-.23-.433-.447a1.834 1.834 0 0 1-.277-.676c-.027-
.307.044-.63.213-.974.17-.342.373-.577.613-.703.17-.072.31-.108.423-.108zM16
3.465C9.088 3.465 3.464 9.088 3.464 16c0 6.913 5.624 12.536 12.536
12.536S28.536 22.913 28.536 16c0-6.912-5.624-12.535-12.536-12.535zM16
1.6c7.94 0 14.4 6.46 14.4 14.4S23.94 30.4 16 30.4 1.6 23.94 1.6 16 8.06 1.6
16 1.6z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.promotion {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M6 4c.4 0 .7.2 1 .4.5.6.5 1.5-.1 2-.5.5-1.4.5-1.9-.1s-
.5-1.4.1-1.9c.2-.3.6-.4.9-.4zm0-1c-.6 0-1.2.2-1.6.6-1 .9-1.1 2.4-.2 3.4.9 1
2.4 1 3.4.2 1-.9 1-2.4.1-3.4C7.3 3.2 6.6 3 6 3zm.3-3l7.1 1 18 19.6L18.9 32
.9 12.4.6 5.3z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.social {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M15.254 19.089c.996 0 3.587 1.992 3.786 8.271-
19.03.199s-.498-7.97 5.579-8.37c2.392-.398 4.783 1.993 6.178 1.793 1.295-
.198 2.59-1.892 3.487-1.892zm3.886-2.69c.797 0 1.793 1.495 2.989 1.694
1.096.198 3.188-1.993 5.18-1.595 5.082.3 4.684 7.573 4.684 7.573l-11.558-
.1c-.697-3.687-2.391-5.181-3.288-5.38.598-1.594 1.495-2.192 1.993-2.192zm-
8.17-9.963c2.79 0 4.98 2.49 4.98 5.679 0 3.089-2.19 5.679-4.98 5.679-2.79 0-
4.982-2.491-4.982-5.68 0-3.088 2.192-5.678 4.982-5.678zm11.657-1.994c2.49 0
4.583 2.293 4.583 5.082 0 2.79-2.092 5.082-4.583 5.082s-4.583-2.293-4.583-
5.082c0-2.79 2.092-5.082 4.583-5.082z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.primary {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M1 7.5h10v2H2v18h28v-18h-9v-2h10a1 1 0 0 1 1 1v20a1 1 0
0 1-1 1H1a1 1 0 0 1-1-1v-20a1 1 0 0 1 1-1zm14-5h2v13.172l2.536-2.536 1.414
1.414L17 18.5l-1 1-1-1-3.95-3.95 1.414-1.414L15 15.672z'/%3E%3C/svg%3E") no-
repeat 100% 100%;
}
.draft {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M10.262 23.448l2.295 2.298-3.443 1.149zm6.886-
6.895l2.296 2.298-5.739 5.746-2.295-2.298zm3.443-3.448l2.295 2.299-2.295
2.298-2.295-2.298zM21 3.414V7h3.785zM6 2v28h20V9h-4.833C20.062 9 19 8.137 19
7.032V2zm.167-2h14.414L28 7.586V30c0 1.103-.73 2-1.833 2h-20C5.064 32 4
31.103 4 30V2C4 .897 5.064 0 6.167 0z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.outbox {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
```

```

fill='%239E9E9E' d='M2 9.63V26h28.001V9.67L15.998 20.08zM2 6v1.134l14.001
10.452 14-10.408L30 6zm0-2h28c1.103 0 2 .897 2 2v20c0 1.103-.897 2-2 2H2c-
1.103 0-2-.897-2-2V6c0-1.103.897-2 2-2z'/%3E%3C/svg%3E") no-repeat 100%
100%;
    }
    .sent {
        background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M6.756 7.527l6.997 8.487-7.02 8.516 20.126-8.457zM0
2.48132 13.603L0.24 29.521l11.135-13.506z'/%3E%3C/svg%3E") no-repeat 100%
100%;
    }
    .important {
        background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M1.3 2.7c.7 0 1.3.6 1.3 1.3v25.2c0 .7-.6 1.3-1.3 1.3-.7
0-1.3-.6-1.3-1.3V4c0-.7.6-1.3 1.3-1.3zm10.3-1.2C18.2 1.5 23.7 5 32 2v17.5c-
11.1 4-17.1-3.7-27.7 1V3.1c2.7-1.2 5-1.6 7.3-1.6z'/%3E%3C/svg%3E") no-repeat
100% 100%;
    }
    .starred {
        background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M16 1.6l4.45 9.48 9.95 1.52-7.2 7.38 1.7 10.42-8.9-4.92-
8.9 4.92 1.7-10.42-7.2-7.38 9.951-1.52z'/%3E%3C/svg%3E") no-repeat 100%
100%;
    }
</style>

```

HOMECONTROLLER.CS

```

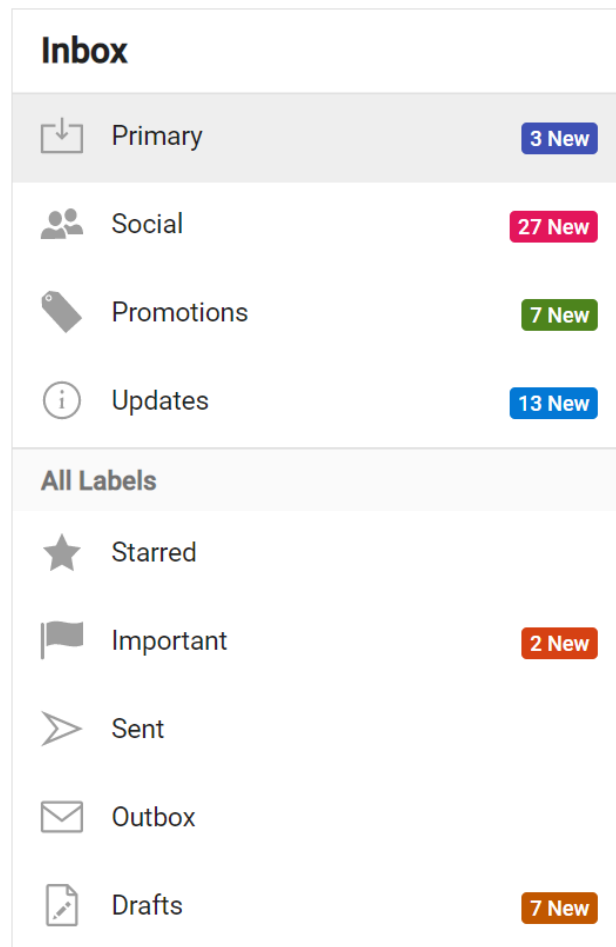
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class BadgeController : Controller
    {
        public IActionResult Listview()
        {
            List<object> data = new List<object>();
            data.Add(new { id = "p_01", text = "Primary", messages = "3
New", badge = "e-badge e-badge-primary", icons = "primary", type = "Primary"
});
            data.Add(new { id = "p_02", text = "Social", messages = "27
New", badge = "e-badge e-badge-secondary", icons = "social", type =
"Primary" });
            data.Add(new { id = "p_03", text = "Promotions", messages = "7
New", badge = "e-badge e-badge-success", icons = "promotion", type =
"Primary" });
            data.Add(new { id = "p_04", text = "Updates", messages = "13
New", badge = "e-badge e-badge-info", icons = "updates", type = "Primary"
});
        }
    }
}

```

```

        data.Add(new { id = "p_05", text = "Starred", messages = "",
badge = "", icons = "starred", type = "All Labels" });
        data.Add(new { id = "p_06", text = "Important", messages = "2
New", badge = "e-badge e-badge-danger", icons = "important", type = "All
Labels" });
        data.Add(new { id = "p_07", text = "Sent", messages = "", badge
= "", icons = "sent", type = "All Labels" });
        data.Add(new { id = "p_08", text = "Outbox", messages = "",
badge = "", icons = "outbox", type = "All Labels" });
        data.Add(new { id = "p_09", text = "Drafts", messages = "7 New",
badge = "e-badge e-badge-warning", icons = "draft", type = "All Labels" });
        ViewBag.dataSource = data;
        return View();
    }
}

```



Note: [View Sample in GitHub.](#)

Dynamic Badge Content

Badges in real-time needs to be updated dynamically based on the requirements. The following sample demonstrates how to update the badges content dynamically. Click the increment button to change the badge value.

CSHTML

```
@using Syncfusion.EJ2
@{
    var template = "<div class=\"listWrapper\" style=\"width: inherit; height: inherit;\>" +
        "<span class=\"${icons} list_svg\"> </span>" +
        "<span class=\"list_text\">${text}</span>" +
        "${if(messages != '')}" +
        "<span class='${badge}' style=\"float: right; margin-top: 16px; font-size: 12px;\>${messages}</span>" +
        "${/if}" +
        "</div>";
}
<div class="col-lg-12 control-section">
    <div class="sample_container badge-list">
        <!-- Listview element -->

@Html.EJS().ListView("lists").Enable(true).DataSource((IEnumerable<object>)ViewBag.dataSource).HeaderTitle("Inbox").ShowHeader(true).Template(template).ActionComplete("actionComplete").Fields(new Syncfusion.EJ2.Lists.ListViewFieldSettings{ GroupBy = "type" }).Render()
        <p class='crossline'></p>
        <span class='incr_button'>
            <button class='e-btn e-primary' id='button'>Increment Badge
Count</button>
        </span>
    </div>
</div>
<style>
    .control-section {
        overflow: auto;
    }
    .sample_container.badge-list {
        max-width: 650px;
        margin: auto;
        height: 540px;
    }
    #lists {
        width: 370px;
        display: inline-block;
        border: 1px solid rgba(0, 0, 0, 0.12)
    }
    #lists ul li:first-child {
        display: none;
    }
    .incr_button {
        vertical-align: top;
        position: relative;
        top: 250px;
        display: inline-block;
    }
</style>
```

```

.crossline {
    display: inline-block;
    height: 100%;
    margin: 0 20px;
    width: 1px;
    background: #8080805c;
}
#lists .e-list-item {
    cursor: pointer;
    height: 50px;
    line-height: 48px;
    border: 0;
}
/* SVG Icons and Customization */
.list_svg {
    width: 24px;
    height: 24px;
    display: inline-block;
    margin-top: 11px;
    margin-right: 16px;
}
.list_text {
    width: 60%;
    display: inline-block;
    vertical-align: top;
}
.updates {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M17.024 13.812l.022.162v.893c0 .487-.007.942-.022 1.366-
.012.423-.027.833-.042 1.23v2.435c0 .415.022.799.064
1.15.042.35.11.559.202.622.091.063.232.095.422.095h.328l.105.324c-.041.073-
.125.136-.253.189-.127.055-.226.063-.296.028H14.126a.556.556 0 0 1-.34-
.109v-.27c.086-.108.22-.162.403-.162h.38c.185-.109.294-.219.328-.328.036-
.108.054-.3.054-.573a7.32 7.32 0 0 0 .042-.819c0-.29.007-.527.021-
.709V17.233c0-.636-.018-1.041-.053-1.213-.034-.173-.082-.386-.138-.64a20.09
20.09 0 0 1-.455.19c-.176.072-.291.108-.348.108l-.105-.162c0-.162.09-
.307.274-.434l.38-.27a6.472 6.472 0 0 1 .847-.406c.227-.144.469-.265.73-
.365.26-.1.554-.176.878-.229zm-.952-5.736a.9.9 0 0 1
.613.243c.184.162.343.361.477.595.135.235.2.451.2.65 0 .451-.129.802-.39
1.054-.261.254-.453.397-.572.434a1.424 1.424 0 0 1-.412.054.934.934 0 0 1-
.455-.122c-.15-.082-.293-.23-.433-.447a1.834 1.834 0 0 1-.277-.676c-.027-
.307.044-.63.213-.974.17-.342.373-.577.613-.703.17-.072.31-.108.423-.108zM16
3.465C9.088 3.465 3.464 9.088 3.464 16c0 6.913 5.624 12.536 12.536
12.536S28.536 22.913 28.536 16c0-6.912-5.624-12.535-12.536-12.535zM16
1.6c7.94 0 14.4 6.46 14.4 14.4S23.94 30.4 16 30.4 1.6 23.94 1.6 16 8.06 1.6
16 1.6z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.promotion {
    background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M6 4c.4 0 .7.2 1 .4.5.6.5 1.5-.1 2-.5.5-1.4.5-1.9-.1s-
.5-1.4.1-1.9c.2-.3.6-.4.9-.4zm0-1c-.6 0-1.2.2-1.6.6-.1 .9-1.1 2.4-.2 3.4.9 1
2.4 1 3.4.2 1-.9 1-2.4.1-3.4C7.3 3.2 6.6 3 6 3zm.3-3l7.1 1 18 19.6L18.9 32
.9 12.4.6 5.3z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.social {

```



```

background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M15.254 19.089c.996 0 3.587 1.992 3.786 8.271-
19.03.199s-.498-7.97 5.579-8.37c2.392-.398 4.783 1.993 6.178 1.793 1.295-
.198 2.59-1.892 3.487-1.892zm3.886-2.69c.797 0 1.793 1.495 2.989 1.694
1.096.198 3.188-1.993 5.18-1.595 5.082.3 4.684 7.573 4.684 7.5731-11.558-
.1c-.697-3.687-2.391-5.181-3.288-5.38.598-1.594 1.495-2.192 1.993-2.192zm-
8.17-9.963c2.79 0 4.98 2.49 4.98 5.679 0 3.089-2.19 5.679-4.98 5.679-2.79 0-
4.982-2.491-4.982-5.68 0-3.088 2.192-5.678 4.982-5.678zm11.657-1.994c2.49 0
4.583 2.293 4.583 5.082 0 2.79-2.092 5.082-4.583 5.082s-4.583-2.293-4.583-
5.082c0-2.79 2.092-5.082 4.583-5.082z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.primary {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M1 7.5h10v2H2v18h28v-18h-9v-2h10a1 1 0 0 1 1 1v20a1 1 0
0 1-1 1H1a1 1 0 0 1-1-1v-20a1 1 0 0 1 1-1zm14-5h2v13.17212.536-2.536 1.414
1.414L17 18.51-1 1-1-3.95-3.95 1.414-1.414L15 15.672z'/%3E%3C/svg%3E") no-
repeat 100% 100%;
}
.draft {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M10.262 23.44812.295 2.298-3.443 1.149zm6.886-
6.89512.296 2.298-5.739 5.746-2.295-2.298zm3.443-3.44812.295 2.299-2.295
2.298-2.295-2.298zM21 3.414V7h3.785zM6 2v28h20V9h-4.833C20.062 9 19 8.137 19
7.032V2zm.167-2h14.414L28 7.586V30c0 1.103-.73 2-1.833 2h-20C5.064 32 4
31.103 4 30V2C4 .897 5.064 0 6.167 0z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.outbox {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M2 9.63V26h28.001V9.67L15.998 20.08zM2 6v1.134114.001
10.452 14-10.408L30 6zm0-2h28c1.103 0 2 .897 2 2v20c0 1.103-.897 2-2 2H2c-
1.103 0-2-.897-2-2V6c0-1.103.897-2 2-2z'/%3E%3C/svg%3E") no-repeat 100%
100%;
}
.sent {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M6.756 7.52716.997 8.487-7.02 8.516 20.126-8.457zM0
2.48132 13.603L0.24 29.52111.135-13.506z'/%3E%3C/svg%3E") no-repeat 100%
100%;
}
.important {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M1.3 2.7c.7 0 1.3.6 1.3 1.3v25.2c0 .7-.6 1.3-1.3-.7
0-1.3-.6-1.3-1.3V4c0-.7.6-1.3 1.3-1.3zm10.3-1.2C18.2 1.5 23.7 5 32 2v17.5c-
11.1 4-17.1-3.7-27.7 1V3.1c2.7-1.2 5-1.6 7.3-1.6z'/%3E%3C/svg%3E") no-repeat
100% 100%;
}
.starred {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 32 32'%3E%3Cpath
fill='%239E9E9E' d='M16 1.614.45 9.48 9.95 1.52-7.2 7.38 1.7 10.42-8.9-4.92-

```

```

8.9 4.92 1.7-10.42-7.2-7.38 9.951-1.52z'/%3E%3C/svg%3E") no-repeat 100%
100%;
}
</style>
<script type="text/javascript">
    var badgeElements;
    function actionComplete() {
        badgeElements =
Array.prototype.slice.call(document.getElementById('lists').getElementsByClassName('e-badge'));
        document.getElementById('button').addEventListener('click', function
buttonClick() {
            badgeElements.forEach((element) => {
                element.textContent = (Number(element.textContent.split('
')[0])) + 1 + ' New';
            });
        });
    }
</script>

```

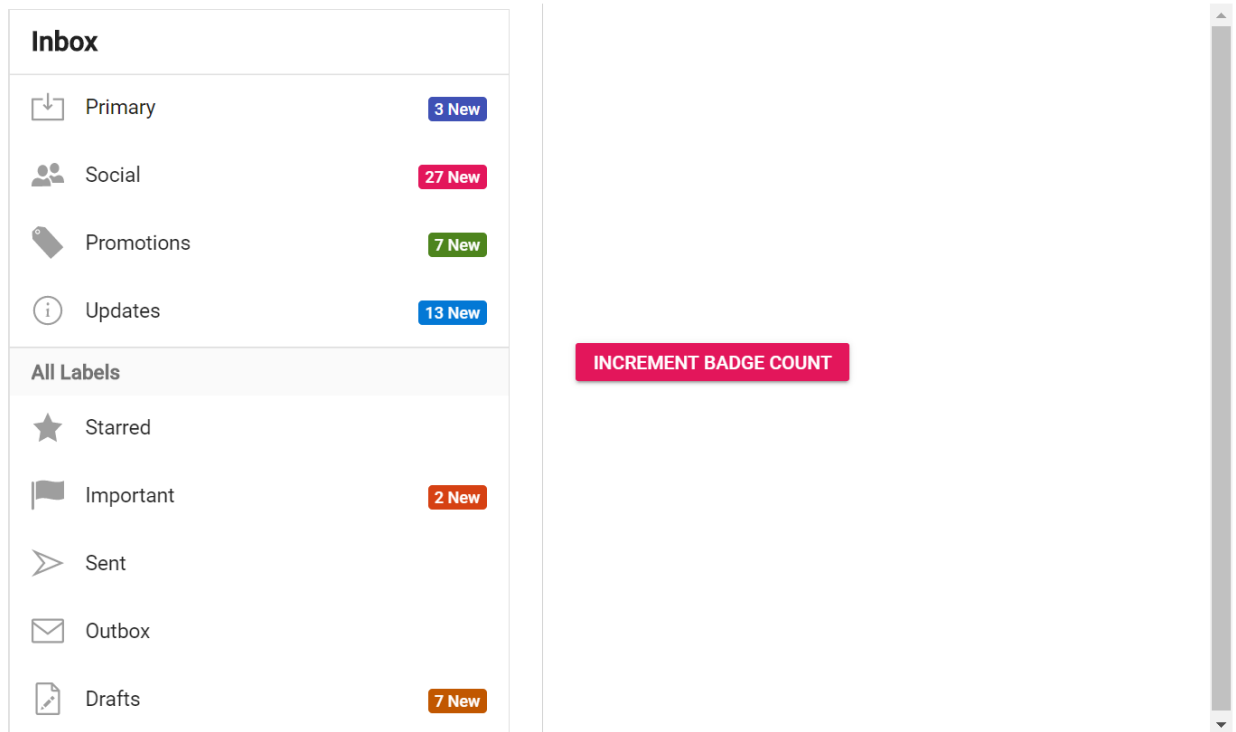
HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class BadgeController : Controller
    {
        public IActionResult Listview()
        {
            List<object> data = new List<object>();
            data.Add(new { id = "p_01", text = "Primary", messages = "3
New", badge = "e-badge e-badge-primary", icons = "primary", type = "Primary"
});
            data.Add(new { id = "p_02", text = "Social", messages = "27
New", badge = "e-badge e-badge-secondary", icons = "social", type =
"Primary" });
            data.Add(new { id = "p_03", text = "Promotions", messages = "7
New", badge = "e-badge e-badge-success", icons = "promotion", type =
"Primary" });
            data.Add(new { id = "p_04", text = "Updates", messages = "13
New", badge = "e-badge e-badge-info", icons = "updates", type = "Primary"
});
            data.Add(new { id = "p_05", text = "Starred", messages = "",
badge = "", icons = "starred", type = "All Labels" });
            data.Add(new { id = "p_06", text = "Important", messages = "2
New", badge = "e-badge e-badge-danger", icons = "important", type = "All
Labels" });
            data.Add(new { id = "p_07", text = "Sent", messages = "", badge
= "", icons = "sent", type = "All Labels" });
            data.Add(new { id = "p_08", text = "Outbox", messages = "",
badge = "", icons = "outbox", type = "All Labels" });

```

```
data.Add(new { id = "p_09", text = "Drafts", messages = "7 New",  
badge = "e-badge e-badge-warning", icons = "draft", type = "All Labels" });  
ViewBag.dataSource = data;  
return View();  
}  
}
```



Note: [View Sample in GitHub.](#)

Barcode

Getting Started with ASP.NET MVC Barcode Control

This section briefly explains about how to include [ASP.NET MVC Barcode](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
```

```
</body>
```

Add ASP.NET MVC Barcode control

Now, add the Syncfusion ASP.NET MVC Barcode control in `~/Home/Index.cshtml` page.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").
Height("150px").
Value("123456789").
Type(Syncfusion.EJ2.BarcodeGenerator.BarcodeType.Codabar).Render())
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Barcode control will be rendered in the default web browser.



Adding QR Generator control

You can add the QR code in our barcode generator component.

CSHTML

```
@(Html.EJS().QRCodeGenerator("container").
Width("200px").
Height("150px").
Value("Syncfusion").
Render())
```



Adding Datamatrix Generator control

You can add the datamatrix code in our barcode generator component.

CSHTML

```
@(Html.EJS().
```

```
DataMatrixGenerator("container").  
Width("200px").Height("150px").  
Value("Syncfusion").  
Render() )
```



Note: [View Sample in GitHub](#)

ASP.NET MVC BarcodeGenerator Control

Code39

The Code 39 character set includes the digits 0-9, the letters A-Z (upper case only), and the symbols: space, minus (-), plus (+), period (.), dollar sign (\$), slash (/), and percent (%). A special start or stop character is placed at the beginning and ending of each barcode. The barcode can be of any length; even more than 25 characters begin to push the bounds. Code 39 is the only type of barcode that does not require a checksum for common use.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").  
Width("200px").  
Height("150px").  
Value("SYNCFUSION").  
Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code39).Render() )
```

CODE39.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using Syncfusion.EJ2.BarcodeGenerator;  
namespace EJ2CoreSampleBrowser.Controllers.Barcode  
{  
    public partial class BarcodeController : Controller  
    {  
        public ActionResult Code39()  
        {  
            return View();  
        }  
    }  
}
```

```
}
```

Code39 Extended

Code 39 Extended is an extended version of Code 39 that supports ASCII character set. In Code 39 Extended, you can also code 26 lower letters (a-z) and the special characters in the keyboard.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").
Height("150px").
Value("SYNCFUSION").
Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code39Extension).Render() )
```

CODE39EXTD.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code39Extension()
        {
            return View();
        }
    }
}
```

Code 11

Code 11 is used primarily for labeling the telecommunication equipment. The character set includes the digits 0 to 9, a dash (-), and a start / stop code.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").
Height("150px").
Value("112").
Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code11).Render() )
```

CODE11.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code11()
        {
            return View();
        }
    }
}
```

Codabar

Codabar is a variable length symbol that encodes the following 20 characters:

0123456789-\$/+.ABCD

The characters, A, B, C and D are used as start and stop characters. Codabar is used in libraries, blood banks, the package delivery industry and a variety of other information processing applications.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").Height("150px").
Value("123456789").Type(Syncfusion.EJ2.BarcodeGenerator.Type.Codabar).Render
() )
```

CODABAR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Codabar()
        {
            return View();
        }
    }
}
```


Code 32

Code 32 is mainly used for coding pharmaceuticals, cosmetics and dietetics. It is often used to encode Italian Pharmacode that has the following structure:

- 'A' character (ASCII 65), that is not really encoded.
- 8 digits for Pharmacode (It generally begins with / and prefixed with 0).
- 1 digit for checksum module 10, that is automatically calculated by barcode.

The value to be encoded must be 8 digits Pharmacode (prefix it with '0' if necessary) and the 9th digit (the checksum) is automatically calculated by barcode.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container")
.Width("200px").
Height("150px").
Value("01234567").Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code32).Render()
)
```

CODE32.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code32()
        {
            return View();
        }
    }
}
```

Code 93

Code 93 is designed to complement and improve upon Code 39. It can represent the entire ASCII character set by using the combinations of 2 characters. Code 93 is a continuous, variable-length symbology and produces denser code. The Standard Mode (default implementation) can encode uppercase letters (A-Z), digits (0-9), and special characters like *, -, \$, %, (Space), ., /, and +.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").
Height("150px").
Value("01234567").
Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code93).Render() )
```

CODE93.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code93()
        {
            return View();
        }
    }
}
```

Code 93 Extended

The Code 93 Extended Barcode symbology is continuous, variable length, and self-checking. It is based on Code 93 Barcode. The Extended Version can encode all 128 ASCII characters.

Code 128

Code 128 is a variable length, high density, alphanumeric, linear bar code symbology, capable of encoding the full 128-character ASCII character set and extended character sets. This symbology includes a checksum digit for verification and the barcode can also be verified character-by-character by verifying the parity of each data byte.

Code 128 Code Sets

- Code Set A (or Chars Set A) includes all of the standard upper case U.S. alphanumeric keyboard characters and punctuation characters along with the control characters (namely, characters with ASCII values from 0 to 95 inclusive), and seven special characters.
- Code Set B (or Chars Set B) includes all of the standard upper case alphanumeric keyboard characters and punctuation characters along with the lower case alphabetic characters (namely, characters with ASCII values from 32 to 127 inclusive), and seven special characters.
- Code Set C (or Chars Set C) includes the set of 100 digit pairs from 00 to 99 inclusive along with three special characters. This allows numeric data to be encoded as two data digits per symbol character, at effectively twice the density of standard data.

Code 128 Special characters

The last seven characters of Code Sets A and B (character values 96 - 102) and the last three characters of Code Set C (character values 100 - 102) are special non-data characters with no ASCII character equivalents that have a particular significance to the Barcode reading device.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").
Height("150px").
Value("SYNCFUSION").Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code128).Rende
r() )
```

CODE128.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code128()
        {
            return View();
        }
    }
}
```

Customizing the Barcode color

A page or printed media with barcode often appears colorful in the background and surrounding region with other contents. In such cases, the barcode can also be customized to suit the needs. You can achieve this by using for forecolor property.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").
Width("200px").
Height("150px").
Value("SYNCFUSION").Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code128).Rende
r() )
```

CODE128.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code128()
```

```
    {  
        return View();  
    }  
}
```

Customizing the Barcode dimension

The dimension of the barcode can be changed using the height and width property of the barcodegenerator.

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").  
Width("300px").  
Height("300px").  
foreColor("red").  
Value("SYNCFUSION").Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code128).Rende  
r() )
```

DIMENSION.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using Syncfusion.EJ2.BarcodeGenerator;  
namespace EJ2CoreSampleBrowser.Controllers.Barcode  
{  
    public partial class BarcodeController : Controller  
    {  
        public ActionResult Code128()  
        {  
            return View();  
        }  
    }  
}
```

Customizing the text

In barcode generators, you can customize the barcode text by using display text property .

CSHTML

```
@(Html.EJS().BarcodeGenerator("container").Width("200px").Height("150px").  
Type(Syncfusion.EJ2.BarcodeGenerator.Type.Code128).  
Value("SYNCFUSION").DisplayText(s =>  
s.text("text")).Invalid("invalidInput").Render() )
```

TEXT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult Code128()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

ASP.NET MVC QR Code generator Control

QR Code

A QR Code is a two-dimensional barcode that consists of a grid of dark and light dots or blocks that form a square. The data encoded in the barcode can be numeric, alphanumeric, or Shift Japanese Industrial Standards (JIS8) characters. The QR Code uses version from 1 to 40. Version 1 measures 21 modules x 21 modules, Version 2 measures 25 modules x 25 modules, and so on. The number of modules increases in steps of 4 modules per side up to Version 40 that measures 177 modules x 177 modules. Each version has its own capacity. By default, the barcode control automatically sets the version according to the length of the input text. The QR Barcodes are designed for industrial uses and also commonly used in consumer advertising.

CSHTML

```
@(Html.EJS().QRCodeGenerator("container").
Width("200px").
Height("150px").
Value("SYNCFUSION").
Render() )
```

QRCODE.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult qrcode()
        {
        }
    }
}
```

```

        {
            return View();
        }
    }
}

```

Customizing the QR Code color

A page or printed media with QR Code often appears colorful in the background and surrounding region with other contents. In such cases, the QR Code can also be customized to suit the needs. You can achieve this by using for forecolor property.

CSHTML

```

@(Html.EJS().QRCodeGenerator("container").
    Width("200px").
    foreColor("red").
    Height("150px").
    Value("SYNCFUSION").Render() )

```

COLOR.CS

```

using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}

```

Customizing the QR Code dimension

The dimension of the QR Code can be changed using the height and width properties of the QR Code Generator.

CSHTML

```

@(Html.EJS().QRCodeGenerator("container").
    Width("300px").
    foreColor("red").
    Height("300px").
    Value("SYNCFUSION").Render() )

```

DIMENSION.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}
```

Customizing the text

In QR Code generators, you can customize the QR Code text by using display text property.

CSSHTML

```
@(Html.EJS().QRCodeGenerator("container").
Width("200px").
foreColor("red").
Height("150px").
DisplayText(s => s.text("text"))).
Value("SYNCFUSION").Render() )
```

TEXT.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Data Matrix generator

Data Matrix

DataMatrix Barcode is a two dimensional barcode that consists of a grid of dark and light dots or blocks forming square or rectangular symbol. The data encoded in the barcode can either be numbers or alphanumeric. They are widely used in printed media such as labels and letters. You can read it easily with the help of a barcode reader and mobile phones.

CSHTML

```
@(Html.EJS().DataMatrixGenerator("container").
Width("200px").
Height("150px").
Value("SYNCFUSION").Render() )
```

DATAMATRIXGENERATOR.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}
```

Customizing the DataMatrix color

A page or printed media with DataMatrix often appears colorful in the background and surrounding region with other contents. In such cases, the DataMatrix can also be customized to suit the needs. You can achieve this by using the forecolor property.

CSHTML

```
@(Html.EJS().DataMatrixGenerator("container").
Width("200px").
foreColor("red").
Height("150px").
Value("SYNCFUSION").Render() )
```

COLOR.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
```



```
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}
```

Customizing the DataMatrix dimension

The dimension of the DataMatrix can be changed using the height and width property of the DataMatrix Generator.

CSHTML

```
@(Html.EJS().DataMatrixGenerator("container").
Width("300px").
foreColor("red").
Height("300px").
Value("SYNCFUSION").Render() )
```

DIMENSION.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}
```

Customizing the text

In DataMatrix generators, you can customize the DataMatrix text by using the display text property.

CSHTML

```
@(Html.EJS().DataMatrixGenerator("container").
Width("200px").
foreColor("red").
Height("150px").
DisplayText(s => s.text("text")).
```

```
Value("SYNCFUSION").Render() )
```

TEXT.CS

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.BarcodeGenerator;
namespace EJ2CoreSampleBrowser.Controllers.Barcode
{
    public partial class BarcodeController : Controller
    {
        public ActionResult datamatrix()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Export

Export barcode as an image and base64 string is common for barcode, QRcode and datamatrix.

Export

Barcode provides the support to export its content as an image in the specified image type and downloads it in the browser. The following code example shows how to export the barcode as an image

`typescript

```
var barcode = document.getElementById("barcode").ej2_instances[0];
var filename = 'Export';
barcode.exportImage(filename, 'JPG');
```

,

The filename specifies the name of the file to be downloaded.

Export As Base64String

Barcode provides the support to export its content as an image in the specified image type and returns it as base64 string. The following code example shows how to export the barcode as a base64 string,

`typescript

```
var barcode = document.getElementById("barcode").ej2_instances[0];
await barcode.exportAsBase64Image('JPG');
```

,

Note: Format is to specify the type or format of the exported file. You can export the barcode to the following formats:

* JPG.

* PNG.

Breadcrumb

Getting Started with ASP.NET MVC Breadcrumb Control

This section briefly explains about how to include [ASP.NET MVC Breadcrumb](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

<add namespace="Syncfusion.EJ2"/>

</namespaces>

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
```

```

...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>

```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```

<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>

```

Add ASP.NET MVC Breadcrumb control

Now, add the Syncfusion ASP.NET MVC Breadcrumb control in `~/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().Breadcrumb("default").Render()
```

Add Items to the Breadcrumb Control

Use [Items](#) property to bind items for Breadcrumb control. The below example demonstrates the basic rendering of Breadcrumb with items support.

CSHTML

```

@Html.EJS().Breadcrumb("default").Items(item =>
{
    item.IconCss("e-icons e-home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

    item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridOverview").Add();

    item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/DefaultFunctionalities").Add();

    item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/DefaultFunctionalities").Add();
}).Render()
<style>

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Breadcrumb control will be rendered in the default web browser.

[Home](#) / [Components](#) / [Navigations](#) / Breadcrumb

Enable or Disable Navigation

This feature enables or disables the item navigation. By default, the navigation will be enabled when setting [Url](#) property. To prevent breadcrumb item navigation, set [EnableNavigation](#) property as false in Breadcrumb. The below example shows enabling and disabling the navigation of Breadcrumb items.

The following example shows a Enable or disable navigation of Breadcrumb.

CSHTML

```
<div id="breadcrumb-control">
    <div class='header'><b>EnableNavigation - false</b></div><br />

    @Html.EJS().Breadcrumb("disable").EnableNavigation(false).Items(item =>
    {
        item.IconCss("e-icons e-home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

        item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridViewOverview").Add();

        item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/DefaultFunctionalities").Add();

        item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/DefaultFunctionalities").Add();
    }).Render()
    <br />
    <br />
    <div class='header'><b>EnableNavigation - true</b></div><br />
    =>
    {
        item.IconCss("e-icons e-home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

        item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridViewOverview").Add();

        item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/DefaultFunctionalities").Add();

        item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/DefaultFunctionalities").Add();
    }).Render()
</div>
<style>
    @@font-face {
        font-family: 'e-bicons';
        src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAIAIAAwAgTlMvMj1wSfkAAAEoAAAVmNtYXNHOdpAAABmAAAAD5nbHlmSRvkRAAAAgAAANoaGVhZB2Xb78AADQAAAAANmhoZWEIuQQAHAARAAAACRobXR4GAAAAAAAAAYAAAAAYbG9jYQSCAv4AAAHYAAAAADmlheHABFwEfAAABCAAAACBuYW1lXj4/wAABVAAAAI1cG9zdE4LDlo
```

```

AAAd4AAAAegABAAAEAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGARMABwAAAAA
AAgAAAAoACgAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAQA AAA
EAAAABAAAAQA AAAEAAAAAAAAAgAAAAAUAAMAAQAAABQABAAqAAAAABAAEAAEAoCH//8AAOc
D//8AAAABAAQAAAAABAIABQADAAQAAAAAAABTgFqAYABLAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAEVDwUrAS8FPQE/BTsBHWUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAychFw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREvHwgzITM/CDURNS8
IIyECGAICAwQEBAUFbQQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAwUFBQcGBwgICakJCQgJCAcHBwY
GBQQEAWIBAQEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwMDQ4ODg0NDAwLCwkJCAcFBQMCohYTeHiIKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAgUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEAWIBAQEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMdBQUBwYHCAGICQkODQ0MDAsLCQkIBwYEAWIBAwMEBgICaOLCwMDQ0ODg0NDQw
LCGoJBwCGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikKRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzCLCwRHTEcDRETEw0JOAKUEBAUKSQpBwgGBQI2fHEt/JQCkC39QwYGBgsKCQYFAGe
BAgUGCQoLBgYGA2wBgYLCgkGBQIBAAACAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAACADwABAAA
AAAAEAACAFgABAAAAAAFAAsAHQABAAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBlLWJjb25
zUmVndWxhcmUtYmNvbNlLWJjb25zVmVyc2lubiAxLjBlLWJjb25zRm9udCBnZW5lcmF0ZWQgdXN
pbmcgU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQBByAHMAaQB
vAG4AIAAxAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIAIBnAGUAbgBlAHIAIYQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAIBNAGUAdABYAG8AIABTAHQAdQkAgkAbwB
3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEAGQcAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hly2stbWFyYXhob3VzZS0wNAAAAA=)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-bicons' !important;
    font-size: 14px;
}
#breadcrumb-control {
    margin-left: auto;
    margin-right: auto;
    width: 60%;
}

#breadcrumb-control .header {
    text-align: left;
    padding-left: 10px;
}

#breadcrumb-control .e-control.e-breadcrumb {
    text-align: left;
}
</style>

```

EnableNavigation - false

 / [Components](#) / [Navigations](#) / Breadcrumb

EnableNavigation - true

 / [Components](#) / [Navigations](#) / Breadcrumb

Note: [View Sample in GitHub.](#)

Data Binding in Breadcrumb

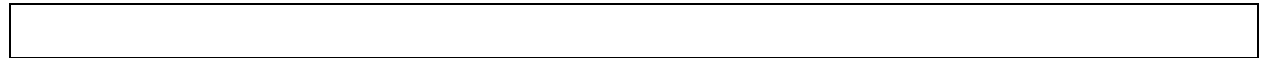
Items based on current Url

The Breadcrumb items can be generated from the current URL of the page, if the `url` property was not provided or when the user does not specify the breadcrumb items using the `BreadcrumbItemDirective` tag. The following example shows the breadcrumb items that are generated based on the current URL.

CSHTML

```
@Html.EJS().Breadcrumb("default")..EnableNavigation(false).Render()
```

CURRENT-URL.CS



Output be like the below.

 / [breadcrumb](#) / bind-to-location?theme=bootstrap5

Note: This output screenshot shows the [Bind to Location](#) sample.

 This sample is hosted in different location, so the breadcrumb is rendered with different location instead of the actual location.

Static URL

The Breadcrumb items can be generated by providing the `url` property in the component, if the user does not specify the breadcrumb items using the `BreadcrumbItemDirective` tag. The following example shows the breadcrumb items generated from the provided url in the component.

CSHTML

```
@Html.EJS().Breadcrumb("url").EnableNavigation(false).Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/DefaultFunctionalities").Render()
```

DEFAULT.CS


```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

 / [aspnetmvc](#) / [Breadcrumb](#) / DefaultFunctionalities

Customize text when generated items using Url

The Breadcrumb items text can be customized by using the `beforeItemRender` event. In the following example, `All Components` text was customized as `Components`.

CSHTML

```
@Html.EJS().Breadcrumb("url").EnableNavigation(false).BeforeItemRender("beforeItemRender").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/BindToLocation").Render()

<script>
    function beforeItemRender(args) {
        if (args.item.text === 'BindToLocation') {
            args.item.text = 'location';
        }
    }
</script>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

 / [aspnetcore](#) / [Breadcrumb](#) / location

Icons

The Breadcrumb contains an icon/image to provide a visual representation of an item.

Loading icon in BreadcrumbItem

To load the icon on the breadcrumb item, set the `iconCss` property.

Breadcrumb with Font Icon

To place the font icon on the breadcrumb item, set the `iconCss` property to `e-icons` with the required icon CSS. By default, the icon is positioned to the left side of the item.

CSHTML

```
@Html.EJS().Breadcrumb("default").Items(item =>
{
    item.IconCss("e-icons e-
home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

    item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridO
verview").Add();

    item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/Defa
ultFunctionalities").Add();

    item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb
/DefaultFunctionalities").Add();
}).Render()
<style>
    @@font-face {
        font-family: 'e-bicons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXNHOdpAAABmAAAAD5nbHlMsrV
kRAAAAEgAAANoaGVhZB2Xb78AAADQAAANmhoZWEIUQQHAAARAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQSCAv4AAAHYAAADm1heHABFwEFAAABCAAAACBuYW11Xj/4/wAABVAAAAILcG9zdE4LDlO
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAABAAAAAGARMABwAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAOckZAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAABAAAAAABAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAA
EAAAABAAAAQAQAAAEAAAAAAGAAAAAUAAMAAQAAABQABAAqAAAABAAEAAEA0cH//8AAOc
D//8AAAABAAQAAABAAIABQADAAQAAAAAABTgFqAYABlAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAEVDwUrAS8FPQE/BTsBHWUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAychFw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREvHwgzITM/CDURNS8
IiYECGAICAwQEBAUFbQQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAwUFBQcGBWgICAKJCQgJCAChBwY
GBQQEAWIBAQEBAGMEBAUGBgCHBwgJCAKJCQgICACGBWUFBQMDAgLeAQIDBQUHCAKJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwwMDQ4ODg0NDawLCwkJCAcFBQMCohYTeHiKiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBABNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAgUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAChBwYGBQQEAWIBAQEBAGMEBAUGBgCHBwgJCAKJCQgICACGBWU
FBQMDAgICAgMDBQUFBwYHCAGICQkODQ0MDAsLCQkIBwYEAWIBAwMEBgcICAoLCwwMDQ0ODg0NDQw
LCGoJBwCGBAQCAQECAwUGBwcJCGoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzclCwwRHTECDRETEw0JOakUEBAUKSQpBwgGBQI2fHEt/JQckC39QwYGBgsKCQYfAgE
BAgUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAxcD9AAFAAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9AOAAAUAAAEEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRMzUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAABAAAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAAAADAAcADwABAAA
AAAAEAACafgABAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBlLWJjb25
zUmVndWxhcmUtYmNvbNlLWJjb25zVmVyc2lubiAxLjBlLWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcgU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQBByAHMAaQB
vAG4AIAAxAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAEQBuAGMAZgBlAHMAaQBvAG4AIAIBNAGUAdABYAG8AIAIBTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAABAAAAAABAAAAA
AAAAAABAAAAAABAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hlY2stbWYyYXVhob3VzZS0wNAAAAA=)
format('trueType');
```

```

        font-weight: normal;
        font-style: normal;
    }
    .e-bicons {
        font-family: 'e-bicons' !important;
        font-size: 14px;
    }
</style>

```

ITEMS.CS

Output be like the below.

 / [Components](#) / [Navigations](#) / Breadcrumb

Breadcrumb with Image

In the Breadcrumb, images can be added for the items using the `iconCss` property. In the following example, an image was added to the breadcrumb item by using the iconCss class `e-image` and specifying height and width.

CSHTML

```

@Html.EJS().Breadcrumb("default").Items(item =>
{
    item.IconCss("e-image-home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

    item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridOverview").Add();

    item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/DefaultFunctionalities").Add();


    item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/DefaultFunctionalities").Add();
}).Render()
<style>
    .e-image-home {
        background-image: url('home.png');
        height: 20px;
        width: 20px;
    }
    .e-svg-home {
        background-image: url('home.svg');
        height: 20px;
        width: 20px;
    }
</style>

```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

 / Components / Navigations / Breadcrumb

Breadcrumb with SVG Image

In the Breadcrumb, SVG image can be added for the items using the `iconCss` property. In the following example, SVG image was added to the breadcrumb item by using the `iconCss` class `e-image` and specifying height and width.

CSHTML

```
@Html.EJS().Breadcrumb("default").Items(item =>
{
    item.IconCss("e-svg-home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

    item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridOverview").Add();

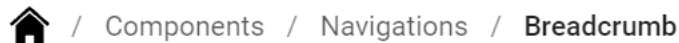
    item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/DefaultFunctionalities").Add();

    item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/DefaultFunctionalities").Add();
}).Render()
<style>
    .e-svg-home {
        background-image: url('home.svg');
        height: 20px;
        width: 20px;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.



Icon Position

By default, the icon is positioned to the left side of the item in the Breadcrumb. If you need to add the icon right to the breadcrumb item, add the `e-icon-right` class to the required item. In the following example, the `e-icon-right` class was added to the breadcrumb items using the `beforeItemRender` event.

CSHTML

```
<div id='loader'>LOADING...</div>
<div id='container'>
  <div id="breadcrumb-control" class="control-section">
    <div class="header"><b>Icon Position - Left</b></div><br />
    @Html.EJS().Breadcrumb("default").Items(item =>
    {
      item.Text("Program Files").IconCss("e-bicons e-folder").Add();
      item.Text("Services").IconCss("e-bicons e-folder").Add();
      item.Text("Config.json").IconCss("e-bicons e-file").Add();
    }).Render()
    <div class="header"><b>Icon Position - Right</b></div><br />

    @Html.EJS().Breadcrumb("icon").BeforeItemRender("beforeItemRender").Items(it
    em =>
    {
      item.Text("Program Files").IconCss("e-bicons e-folder").Add();
      item.Text("Services").IconCss("e-bicons e-folder").Add();
      item.Text("Config.json").IconCss("e-bicons e-file").Add();
    }).Render()
    </div>
  </div>
  <script>
    function beforeItemRender(args) {
      if (args.item.text !== '') {
        args.element.classList.add('e-icon-right');
      }
    }
  </script>
  <style>
    @@font-face {
      font-family: 'e-bicons';
      src: url(data:application/x-font-ttf;charset=utf-
      8;base64,AAEAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXNHOdpAAABmAAAAD5nbHlmSRv
      kRAAAAgAAANoaGVhZB2Xb78AADQAAANmhoZWEIUAQAAAArAAAACRobXR4GAAAAAAAYAAAAA
      YbG9jYQSCAv4AAAHYAAADm1heHABFwEfAAABCAAAACBuYW11Xj/4/wAABVAAAAILcG9zdE4LDlo
      AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAABgABAAAAQAA6q1k4F8
      PPUACwQAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGARMABwAAAAA
      AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
      AAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
      EAAAAABAAAAAQAAAAEAAAAAAAAAagAAAAAMAAAAUAAMAAQAAABQABAAqAAAABAAEAAEAOC//8AAOC
```

```

D//8AAAABAAQAAAABAAIABQADAAQAAAAAAAABTgFqAYABlAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAEVDwUrAS8FPQE/BTsBHWUHFR8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAycHFW8EJwc
fBACXNx8EBxc3HWE/Ahc3Jz8DFzcnPwUnBy8DNycHLwQ1JyM/ASERIREzJREVHwgzITM/CDURNS8
IIyECGAICAwQEBAUFbQQDAwMBAQMDAwQFBQQUEBAQDAgJvAgIDAwUFBQcGBwgICakJCQgJCAcHBwY
GBQQEAWIBAQEBAgMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDbQUHCakJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBagMFBgYICQoKCwwMDQ4ODg0NDAwLCwkJCACFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUdW8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEAWIBAQEBAgMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAGICQkODQ0MDAsLCQkIBwYEAwIBAwMEBgICaOLCwwMDQ0ODg0NDQw
LCgoJBwCGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzcLCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHet/JQCkC39QwYGBgsKCQYFAGe
BAgUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrGcaf4uOv7dtAG9/kMB8Sh/AAAAAAEAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAEAAAAABAAAAAABAACAAQABAAAAAAACAACAABAAAAAADAACADwABAAAA
AAAAEAACAFgABAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBlLWJjb25
zUmVndWxhcmUtYmNvbNlLWJjb25zVmVyc2lvbiAxLjBlLWJjb25zRm9udCBnZW5lcmF0ZWQgdXN
pbmcgU3luY2Zlc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQBhAHMAaQB
vAG4AIAAxAAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBuAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hlY2stbWFyYXhob3VzZS0wNAAAAA=)
format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-bicons' !important;
    font-size: 14px;
}
.e-folder:before {
    content: "\e704";
}
.e-file:before {
    content: "\e703";
}
#breadcrumb-control {
    margin-left: auto;
    margin-right: auto;
    width: 60%;
}
#breadcrumb-control .header {
    text-align: left;
    padding-left: 10px;
}
#breadcrumb-control .e-control.e-breadcrumb {
    text-align: left;
}
</style>

```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

Icon Position - Left

 Program Files /  Services /  Config.json

Icon Position - Right

Program Files  / Services  / Config.json 

Icon Only

To display only icons for the items, add icons using the `iconCss` property. In the following example, breadcrumb items were demonstrated with only icons by providing the `iconCss` property.

CSHTML

```
@Html.EJS().Breadcrumb("default").Items(item =>
{
    item.IconCss("e-icons e-home").Add();
    item.IconCss("e-bicons e-folder").Add();
    item.IconCss("e-bicons e-file").Add();
}).Render()
<style>
@@font-face {
    font-family: 'e-bicons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAVmNtYXNHOdpAAABmAAAAD5nbHlmSRv
kRAAAAEgAAANoaGVhZB2Xb78AADQAAANmhoZWEIUQQHAAArAAAACRobXR4GAAAAAAAYAAAA
YBg9jYQSCAv4AAAHYAAAAADm1heHABFwEfAAABCAAAACBuYw1lXj/4/wAABVAAAAILcG9zdE4LDlo
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAABgABAAAAQA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAGARMABwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAQAAAA
EAAAABAAAAQAAAAEAAAAAAAAAagAAAAAMAAAAUAMAAQAAABQABAAqAAAABAAEAAEOCH//8AAOc
D//8AAAABAAQAAAAABAIABQADAAQAAAAAAAAABTgFqAYABlAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAEVDwUrAS8FPQE/BTsBHWUHFR8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAyCHfw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREVHwgzITM/CDURNS8
IIyECGAICAwQEBAUFBBQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAUFBBQcGBwgICakJCQgJCAChBwY
GBQQEAWIBAQBAGMEBAUGBgCHBwgJCAkJCQgICACGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwwMDQ4ODg0NDAwLCwkJCAcFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
```

```
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEawIBAQBEBagMEBAUGBgchBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAgICQkODQ0MDAsLCQkIBwYEAwIBAwMEBgcICAoLCwwMDQ00Dg0NDQw
LCgoJBwcGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzcLCwwRHTEcDRETEw0JOAkUEBAUKSQpBwgGBQI2fHEt/JQCkC39QwYGBgsKCQYFAgE
BAgUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAATCQEXCQHAcn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAEAAAAABAAAAAABAAcAAQABAAAAAAACAACAABAAAAAADAAcADwABAAA
AAAAEAACAFgABAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyB1LWJjb25
zUmVndWxhcmUtYmNvbnNlLWJjb25zVmVyc2lubiAxLjB1LWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcmU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQByAHMAaQB
vAG4AIAAxAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAIBNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAA
AAAAAAAAAAAAAAAAAYBAgEDAQQBBQEGAQcAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hlY2stbWVfawhob3VzZS0wNAAAAA=)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-bicons' !important;
    font-size: 14px;
}
</style>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.



Show icon only for first item

To show icon only for the first item in the Breadcrumb, add icons to the first item using the `iconCss` property. In the following example, the icon was provided only for the first item by setting the `iconCss` property.

CSHTML

```
@Html.EJS().Breadcrumbs("default").Items(item =>
{
```



```

        item.IconCss("e-icons e-
home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridO
verview").Add();

item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/Defa
ultFunctionalities").Add();

item.Text("Breadcrumb").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb
/DefaultFunctionalities").Add();
}).Render()
<style>
    @@font-face {
        font-family: 'e-bicons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXNHOdpAAABmAAAAD5nbHl1SRv
kRAAAAEgAAANoaGVhZB2Xb78AAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQSCAv4AAAHYAAAAADm1heHABFwEFAAABCAAAACBuYW11Xj/4/wAABVAAAAI1cG9zdE4LDl0
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGARMABwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAaZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAAMAAUAAMAAQAAABQABAAqAAAAABAAEAAEA0cH//8AAOc
D//8AAAAABAAQAAAAABAIABQADAAQAAAAAAABTgFqAYABLAG0AAAABwAAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAAEVDwUrAS8FPQE/BTsBHWUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAYcHFw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREvHwgZITM/CDURNS8
I1yECGAICAwQEBAUFBBQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAUFBBQcGBWgICAKJCQgJCAcHBWY
GBQQEAWIBAQEBAgMEBAUGBgCHBwgJCAKJCQgICACGBWUFBQMDAgLeAQIDBQUHCAKJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAgMFBgYICQoKCwwMDQ4ODg0NDAwLCwkJCAcFBQMCohYTeHiKiYIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDACLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBMDAwEBAwMDBAUFCQgJCAcHBWYGBQQEAWIBAQEBAgMEBAUGBgCHBwgJCAKJCQgICACGBWU
FBQMDAgICAgMDBQUFBWYHCAgICQkODQ0MDAsLCQkIBwYEAWIBAwMEBgICAOcLcwwMDQ00Dg0NDQw
LCgoJBwcGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzclCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHEt/JQCkC39QwYGBgSKCQYFAgE
BAGUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPzA0wAAwALAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAg9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAACADwABAAA
AAAAEAACAFgABAAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBlLWJjb25
zUmVndWxhcmUtYmNvbNlLWJjb25zVmVyc2lubiAxLjBlLWJjb25zRm9udCBnZW5lcmF0ZWQgdXN
pbmcgU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQBByAHMAaQB
vAG4AIAAxAAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIAABnAGUAbgBlAHIAIAYQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBBAAGMAZgBlAHMAaQBvAG4AIAIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBBAAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2Y0Zm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hlY2stbWFyYXhob3VzZS0wNAAAAA=)
format('trueType');
        font-weight: normal;
        font-style: normal;
    }
    .e-bicons {

```

```
font-family: 'e-bicons' !important;
font-size: 14px;
}
</style>
```

ITEMS.CS

Output be like the below.

 / [Components](#) / [Navigations](#) / Breadcrumb

Icons in Breadcrumb

Navigation

The Breadcrumb item navigates to the path while clicking the item. To enable navigation, `url` property was bound to the items.

URL

In the Breadcrumb, the item represents the url. The breadcrumb items can be provided with either relative or absolute URL.

Relative URL

The Breadcrumb items with relative URL contain only the path but do not locate the path or server. The following example represents the breadcrumb items with relative url.

CSHTML

```
@Html.EJS().Breadcrumb("separatorTemplate").EnableNavigation(false).Items(items =>
{
    item.Text("Home").Url("../").Add();
    item.Text("Getting").Url("./breadcrumb/getting-started").Add();
    item.Text("Icons").Url("./breadcrumb/icons").Add();
    item.Text("Navigation").Url("./breadcrumb/navigation").Add();
    item.Text("Overflow").Url("./breadcrumb/overflow").Add();
}).Render()
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

[Home](#) / [Getting](#) / [Icons](#) / [Navigation](#) / Overflow

Absolute URL

The Breadcrumb items with absolute URL contain the path and locate to the resource if the static url is bound to the breadcrumb item. The following example represents the breadcrumb items with static url.

CSHTML

```
@Html.EJS().Breadcrumb("separatorTemplate").EnableNavigation(false).Items(item =>
{
    item.Text("Home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();

    item.Text("Getting").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/grid/overview/").Add();

    item.Text("Icons").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/BindToLocation").Add();

    item.Text("Navigation").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();

    item.Text("Overflow").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/menu/default").Add();
}).Render()
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

[Home](#) / [Getting](#) / [Icons](#) / [Navigation](#) / Overflow

Enable navigation for last Breadcrumb item

The feature enables the last item of the Breadcrumb component by setting the `enableActiveItemNavigation` property to true. In the following example, the last item of the Breadcrumb was enabled.

CSHTML

```
@Html.EJS().Breadcrumb("default").EnableNavigation(false).EnableActiveItemNavigation(true).Items(item =>
{
    item.Text("Home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();
    item.Text("Getting").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/grid/overview/").Add();
    item.Text("Icons").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/BindToLocation").Add();
    item.Text("Navigation").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();
    item.Text("Overflow").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/menu/default").Add();
}).Render()
```

```
item.Text("Home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();

item.Text("Getting").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/grid/overview/").Add();

item.Text("Icons").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/BindToLocation").Add();

item.Text("Navigation").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();

item.Text("Overflow").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/menu/default").Add();
}).Render()
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

Home / Getting / Icons / Navigations / **Overflow**

Open URL in new page or tab

To open the Breadcrumb item in a new page or tab, set the target property of the required item url to blank in the Breadcrumb component. In the following example, the target property of **All Component** item url was set to blank by using the **beforeItemRender** event which locates to the path in the new tab.

CSHTML

```
@Html.EJS().Breadcrumb("separatorTemplate").BeforeItemRender("beforeItemRender").Items(item =>
{
    item.Text("Home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();

    item.Text("Getting").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/grid/overview/").Add();

    item.Text("Icons").Url("https://ej2.syncfusion.com/aspnetmvc/Breadcrumb/BindToLocation").Add();
}
```

```
item.Text("Navigation").Url("https://ej2.syncfusion.com/home/aspnetmvc.html#platform").Add();

item.Text("Overflow").Url("https://ej2.syncfusion.com/aspnetmvc/demos/#/material/menu/default").Add();
}).Render()
<script>
    function beforeItemRender(args) {
        if (args.element.children[0]) {
            args.element.children[0].target = "_blank";
        }
    }
</script>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

Home / Getting / Icons / Navigations / **Overflow**

Overflow Mode in breadcrumb

Overflow Mode

In the Breadcrumb component, `maxItems` and `overflowMode` properties were used to limit the number of breadcrumb items to be displayed.

In the following example, the `maxItems` is set as 3 with `overflowMode` as Default. To prevent breadcrumb item navigation, the `enableNavigation` property has been set to false in the Breadcrumb component.

The following overflow modes are available in the Breadcrumb component.

- Collapsed
- Menu
- Wrap
- Scroll
- Hidden
- None

Collapsed

Collapsed mode shows the first and last Breadcrumb items and hides the remaining items with a collapsed icon. When the collapsed icon is clicked, all items become visible and navigable.

CSHTML

```
@Html.EJS().Breadcrumb("collapsed-
mode").MaxItems(3).EnableNavigation(false).OverflowMode(Syncfusion.EJ2.Navig
ations.BreadcrumbOverflowMode.Collapsed).Items(item =>
{
    item.Text("Home").Url("/").Add();
    item.Text("Breadcrumb").Url("/breadcrumb").Add();
    item.Text("Default").Url("/breadcrumb/default-functionalities").Add();
    item.Text("Icons").Url("/breadcrumb/icons").Add();
    item.Text("Navigation").Url("/breadcrumb/navigation").Add();
    item.Text("Overflow").Url("/breadcrumb/overflow").Add();
}).Render()
<style>
@@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXNHOdpAAABmAAAAAD5nbHlmSRv
kRAAAAEgAAANoAGVhZB2Xb78AAADQAAANmhoZWEIUIUQQAHAARAAACRobXR4GAAAAAAAYAAAA
YbG9jYQSCAv4AAAHYAAAAADm1heHABFwEFAAABCAAAACBuYW11Xj/4/wAABVAAAAILcG9zdE4LDlo
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAABgABAAAAQAAG6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGARMABwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAAABAAAAQAAAAEAAAAAAAAAagAAAAAAAUUAAAAQAABQABAAqAAAABAAEAAEAOC//8AAOC
D//8AAAAABAAQAAAAABAAIABQADAAQAAAAAAABTgFqAYABLAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAEVDwUrAS8FPQE/BTSBHwUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPayCHFw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREVHwgZITM/CDURNS8
IIyECGAICAwQEBAUFBBQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAUFBBQcGBwgICakJCQgJCAChBwY
GBQQEAWIBAQBAGMEBAUGBgCHBwgJCAkJCQgICACGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwwMDQ4ODg0NDawLCwkJCAcFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAChBwYGBQQEAWIBAQBAGMEBAUGBgCHBwgJCAkJCQgICACGBwU
FBQMDAgICAgMDBQUFBwYHCAgICQkODQ0MDAsLCQkIBwYEAWIBAwMEBgICAoLCwwMDQ0ODg0NDQw
LCgoJBwcGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzclCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHEt/JQCkC39QwYGBgSKCQYFAGe
BAGUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRMzUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAEAAAAABAAAAAABAACAAQABAAAAAAACAACAABAAAAAADAAcADwABAAA
AAAAEAACAFgABAAAAAAFAAsAHQABAAAAAAAGAACAABAAAAAAAKACwALwABAAAAAAALABIAWwA
DAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBiyB1LWJjb25
zUmVndWxhcmUtYmNvbN1LWJjb25zVmVyc2lvbiAxLjB1LWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQByAHMAaQB
vAG4AIAAxAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAIBNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAOAAAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hlY2stbWFyYXhob3VzZS0wNAAAAA=)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
```

```
.e-bicons {
    font-family: 'e-icons' !important;
    font-size: 14px;
}
</style>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

Home / ... / Overflow

Menu

Menu mode shows the number of Breadcrumb items that can be accommodated within the container space and creates a submenu with the remaining items.

CSHTML

```
@Html.EJS().Breadcrumb("menu-
mode").MaxItems(3).EnableNavigation(false).OverflowMode(Syncfusion.EJ2.Navig
ations.BreadcrumbOverflowMode.Menu).Items(item =>
{
    item.Text("Home").Url("/").Add();
    item.Text("Breadcrumb").Url("/breadcrumb").Add();
    item.Text("Default").Url("/breadcrumb/default-
functionalities").Add();
    item.Text("Icons").Url("/breadcrumb/icons").Add();

    item.Text("Navigation").Url("/breadcrumb/navigation").Add();

    item.Text("Overflow").Url("/breadcrumb/overflow").Add();
}).Render()
<style>
@@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXNHOdpAAABmAAAAD5nbHlmSRv
kRAAAAEgAAANoaGVhZB2Xb78AAADQAAANmhoZWEIUQQHAAArAAAACRobXR4GAAAAAAYAAAA
YbG9jYQSCAv4AAAHYAAADm1heHABFwEfAAABCAAAACBuYW11Xj/4/wAABVAAAAILcG9zdE4LDlo
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAABgABAAAAQAa6q1k4F8
PPPUACwQAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAGARMABwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAAMAAAUAMAAQAAABQABAAqAAAAABAAEAAEAACoH//8AAOc
D//8AAAABAAQAAAAABAIABQADAAQAAAAAAAAABTgFqAYABLAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAEVDwUrAS8FPQE/BTsBHwUHFR8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAychFW8EJwc
fBAcXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyChLwQ1JyM/ASERIREzJREVHwgzITM/CDURNS8
IIyECGAICAwQEBAUFBBQDawMBAQMDAwQFBQQUEBAQDAgJvAgIDAUFBBQcGBwgICAKJCQgJCAcHBwY
```

```

GBQQEAWIBAQEBAgMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAgMFBgYICQoKCwwMDQ4ODg0NDAwLCwkJCAcFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAgUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEAWIBAQEBAgMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAgICQkODQ0MDAsLCQkIBwYEAWIBAwMEBgICAOlCwwMDQ0ODg0NDQw
LCgoJBwCGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzclCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHet/JQCKc39QwYGBgsKCQYFagE
BAgUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPza0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9AOAAAUAAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRMzUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAEEEEAAAAABAAAAABAAcAAQABAAAAAACAACAAABAAAAAADAACADwABAAAA
AAAAEAACAFgABAAAAAAFAAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyB1LWJjb25
zUmVndWxhcmUtYmNvbN1LWJjb25zVmVyc2lvbiAxLjB1LWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQByAHMAaQB
vAG4AIAAxAAC4AMABlAC0AYgBjAG8AbgBzAEYABwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hly2stbWFyYXhob3VzZS0wNAAAAA=)
format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-icons' !important;
    font-size: 14px;
}
</style>

```

DEFAULT.CS

```

public ActionResult Index()
{
    return View();
}

```

Output be like the below.

Home / ... / Navigation / Overflow

Wrap

Wrap mode wraps the items to multiple lines when the Breadcrumb's width exceeds the container space.

CSHTML

```

@Html.EJS().Breadcrumb("wrap-
mode").MaxItems(3).EnableNavigation(false).OverflowMode(Syncfusion.EJ2.Navig
ations.BreadcrumbOverflowMode.Wrap).Items(item =>

```



```

{
    item.Text("Home").Url("./").Add();
    item.Text("Breadcrumb").Url("./breadcrumb").Add();
    item.Text("Default").Url("./breadcrumb/default-functionalities").Add();
    item.Text("Icons").Url("./breadcrumb/icons").Add();
    item.Text("Navigation").Url("./breadcrumb/navigation").Add();
    item.Text("Overflow").Url("./breadcrumb/overflow").Add();
}).Render()
<style>
@@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMj1wSfkAAAEoAAAVmNtYXDNHODpAAABmAAAAD5nbHlmsRv
kRAAAAgAAANoaGVhZB2Xb78AADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQSCAv4AAAHYAAAAADm1heHABFwEFAAABCAAAACBuYW1lXj/4/wAABVAAAAI1cG9zdE4LDlo
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAABgABAAAAQAA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZWAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAGARMABwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAMAAAAUAMAAQAAABQABAAqAAAABAAEAAEAAOch//8AAOc
D//8AAAABAAQAAAAABAAIABQADAAQAAAAAAABTgFqAYABlAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAEVDwUrAS8FPQE/BTsBHWUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAychFW8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREVHwgzITM/CDURNS8
IIyECGAICAwQEBAUFBBQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAwUFBQcGBwgICakJCQgJCAChBwY
GBQQEAwIBAQBEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwMDQ4ODg0NDawLCwkJCACFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBABEBAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAChBwYGBQQAawIBAQBEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAGICQkODQ0MDASLCQkIBwYEAwIBawMEBgcICAoLCwMDQ0ODg0NDQw
LCgoJBwCGBAQCAQECawUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzcLCwwRHTECDRETEw0JOAKUEBAUKSqpBwgGBQI2fHET/JQckC39QwYGBgsKCQYfAgE
BAGUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRMzUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAEAAAAABAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAADAAcAdwABAAA
AAAAEAACAfGABAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4
AmQADAAEECQAFABYAPwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyB1LWJjb25
zUmVndWxhcmUtYmNvbnN1LWJjb25zVmVyc21vbiAxLjB1LWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcgU3luY2Z1c21vbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c21vbi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQBByAHMAaQB
vAG4AIAAxAAC4AMABlAC0AYgBjAG8AbgBzAEYABwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAEQBuAGMAZgB1AHMAaQBvAG4AIAIABNAGUAdABYAG8AIAIBTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAEQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2h1Y2stbWFyYXhob3VzZS0wNAAAAA=)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-icons' !important;
    font-size: 14px;
}
</style>

```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

Home / Breadcrumb / Default
 / Icons / Navigation
 / Overflow

Scroll

Scroll mode shows an HTML scroll bar when the Breadcrumb's width exceeds the container space.

CSHTML

```
@Html.EJS().Breadcrumb("scroll-
mode").MaxItems(3).EnableNavigation(false).OverflowMode(Syncfusion.EJ2.Navig
ations.BreadcrumbOverflowMode.Scroll).Items(item =>
{
    item.Text("Home").Url("./").Add();
    item.Text("Breadcrumb").Url("./breadcrumb").Add();
    item.Text("Default").Url("./breadcrumb/default-functionalities").Add();
    item.Text("Icons").Url("./breadcrumb/icons").Add();
    item.Text("Navigation").Url("./breadcrumb/navigation").Add();
    item.Text("Overflow").Url("./breadcrumb/overflow").Add();
}).Render()
<style>
@@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXNHOdpAAABmAAAAD5nbHlmSRv
kRAAAAEgAAANoAGVhZB2Xb78AAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQSCAv4AAAHYAAAADm1heHABFwEFAAABCAAAACBuYW1lXj/4/wAABVAAAAILcG9zdE4LDlo
AAAd4AAAAAgABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAABgABAAAAQAA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGARMABwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAAMAAUAAMAAQAAABQABAAqAAAABAAEAAEAOC//8AAOC
D//8AAAABAAQAAAAABAIABQADAAQAAAAAABTgFqAYABLAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAEVDwUrAS8FPQE/BTsBHWUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAYcHFw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREvHwgzITM/CDURNS8
IIyECGAICAwQEBAUFBBQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAUFBBQcGBwgICakJCQgJCAcHBwY
GBQQEAwIBAQEBAgMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAgMFBgYICQoKCwMDQ4ODg0NDawLCwkJCAcFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAgUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEAwIBAQEBAgMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAgICQkODQ0MDAsLCQkIBwYEAwIBAwMEBgcICAoLCwwMDQ0ODg0NDQw
```

```

LCgoJBwcGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkKRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzcLCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHEt/JQCkC39QwYGBgsKCQYFAgE
BAgUGCQoLBgYGA2wBgYLCgkGBQIBAAACAAAAAAPzA0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9AOAAAUAAAEEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAAAEAAAAABAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAAcADwABAAA
AAAAEAACAFgABAAAAAAFAAASAHQABAAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyB1LWJjb25
zUmVndWxhcmUtYmNvbNlLWJjb25zVmVyc2lvbiAxLjB1LWJjb25zRm9udCBnZW5lcmF0ZWQgdXN
pbmcgU3luY2Zlc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAyBvBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQByAHMAaQB
vAG4AIAAxAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAOAAAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hly2stbWFyYXhob3VzZS0wNAAAAA=)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-icons' !important;
    font-size: 14px;
}
</style>

```

DEFAULT.CS

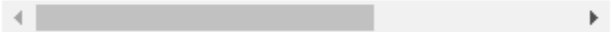
```

public ActionResult Index()
{
    return View();
}

```

Output be like the below.

Home / Breadcrumb / Default / Icons



Hidden

Hidden mode shows the maximum number of items possible in the container space and hides the remaining items. Clicking on a previous item will make the hidden item visible.

CSHTML

```

@Html.EJS().Breadcrumb("default-
mode").MaxItems(3).EnableNavigation(false).OverflowMode(Syncfusion.EJ2.Navig
ations.BreadcrumbOverflowMode.Hidden).Items(item =>
{
    item.Text("Home").Url("./").Add();
    item.Text("Breadcrumb").Url("./breadcrumb").Add();
    item.Text("Default").Url("./breadcrumb/default-
functionalities").Add();
}

```

```

        item.Text("Icons").Url("./breadcrumb/icons").Add();

item.Text("Navigation").Url("./breadcrumb/navigation").Add();

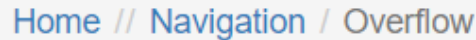
item.Text("Overflow").Url("./breadcrumb/overflow").Add();
    }).Render()
<style>
@@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAVmNtYXNHOdpAAABmAAAAD5nbHlmsRv
kRAAAAEgAAANoaGVhZB2Xb78AAADQAAAAANmhoZWEIUQQHAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQSCAv4AAAHYAAAAADm1heHABFWEfAAABCAAAACBuYw1lXj/4/wAABVAAAAI1cG9zdE4LDlo
AAAd4AAAAegABAAAEAAAAAFWEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAABAAAAAGARMABwAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAagAAAAAUAAMAAQAAABQABAAqAAAABAAEAAEAOC//8AAOC
D//8AAAABAAQAAAAABAIABQADAAQAAAAAABTgFqAYABlAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAEVDwUrAS8FPQE/BTsBHWUHF8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAYcHFW8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREVHWgzITM/CDURNS8
IIyECGAICAwQEBAUFbQQDAwMBAQMDAwQFBQQUEBAQDAgJvAgIDAwUFBQcGBWgICakJCQgJCAcHBwY
GBQQEAWIBAQEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwwMDQ4ODg0NDawLCwkJCACFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUdW8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEAWIBAQEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAGICQkODQ0MDAsLCQkIBwYEAWIBawMEBgcICaOLCwwMDQ0ODg0NDQw
LCgoJBwCGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzCLCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHet/JQCkC39QwYGBgsKCQYfAgE
BAGUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAAPza0wAAwALAAA3IRMhAzMTITUhJyFSauq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAAEAAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAQAAAAAD9A0AAAUAaEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rnEBUQA
AABIA3gABAAAAAABAAAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAAAADAACADwABAAA
AAAAEAACAfGABAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAABKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBiYb1LWJjb25
zUmVndWxhcmUtYmNvbnNlLWJjb25zVmVyc2lubiAxLjB1LWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcgU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQByAHMAaQB
vAG4AIAAxAAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkABwB
3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAABAAAAAABAAAAA
AAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2h1Y2stbWFyYXhob3VzZS0wNAAAAA=)
format('true-type');
    font-weight: normal;
    font-style: normal;
}
.e-bicons {
    font-family: 'e-icons' !important;
    font-size: 14px;
}
</style>

```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.



None

None mode shows all the items on a single line.

Templates in Breadcrumb

The Breadcrumb provides a way to customize the items using `itemTemplate` and the separators using `separatorTemplate` properties.

Item Template

In the following example, Shopping Cart details are used as breadcrumb Items and the items are customized by the chips component using `itemTemplate`.

CSHTML

```
@Html.EJS().Breadcrumb("itemTemplate").ItemTemplate("#chipTemplate").CssClass("e-breadcrumb-chips").Items(item =>
{
    item.Text("Cart").Add();
    item.Text("Billing").Add();
    item.Text("Shipping").Add();
    item.Text("Payment").Add();
}).Render()
<script id="chipTemplate" type="text/x-template">
    <div class="e-lib e-chip-list e-control e-chip-set">
        <div class="e-chip" tabindex="0" role="option" aria-label="Apple"
        aria-selected="false">
            <span class="e-chip-text">${text}</span>
        </div>
    </div>
</script>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.



Separator Template

In the following example, the separators are customized with icons using `separatorTemplate`.

CSHTML

```
@Html.EJS().Breadcrumb("separatorTemplate").SeparatorTemplate("<span
class=\"e-icons e-bullet-arrow\"></span>").Items(item =>
{
    item.Text("Cart").Add();
    item.Text("Billing").Add();
    item.Text("Shipping").Add();
    item.Text("Payment").Add();
}).Render()
<style>
.e-bullet-arrow::before {
    content: '\e253';
    font-size: 21px;
}
@@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1wSfkAAAEoAAAAVmNtYXDNHODpAAABmAAAAD5nbHlmSRv
kRAAAAEgAAANoaGVhZB2Xb78AADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YBg9jYQSCAv4AAAHYAAAAADm1heHABFwEfAAABCAAAACBuYW11Xj/4/wAABVAAAAI1cG9zdE4LDlo
AAAd4AAAAegABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQA6q1k4F8
PPPUACwQAAAAAAN1ClWcAAAAA3UKVZwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGARMABwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQA2AABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBwQAAAAAXAQAAAAAAAABAAAAAABAAAAAQA AAA
EAAAABAAAAAQA AAAEAAAAAAAAAgAAAAAUAAMAAQAAABQABAAqAAAABAAEAAEAOC//8AAOC
D//8AAAABAAQAAAAABAAIABQADAAQAAAAAAABTgFqAYABLAG0AAAABwAAAAADdwP0AB8AXwCfAOM
A5gDsARIAAAEVDwUrAS8FPQE/BTsBHWUHFR8OPw8vDisBDw0XDw8jLw4/Dx8OJxUPAychFw8EJwc
fBACXNx8EBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyCHLwQ1JyM/ASERIREzJREvHwgZITM/CDURNS8
IIyECGAICAwQEBAUFBBQDAwMBAQMDAwQFBQUEBAQDAgJvAgIDAwUFBQcGBwgICakJCQgJCAcHBwY
GBQQEAWIBAQBEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwUFBQMDAgLeAQIDBQUHCAkJCwsMDA0NDg4
ODQwMCwoKCQgGBgUDAgEBAGMFBgYICQoKCwMDQ4ODg0NDawLCwkJCAcFBQMCohYTEhIiKyIOBQo
IBDQJNAEDBQYvHDANDg8IDBQ0FBQUDw8IFDQTEg8NEDAcLwUFBAEBNAo0BwgKECIqIg0RERMLuHF
xPgGW/ZDa/ucBAGUGCQoLBgYHAnAHBgYLCgkGBQIBAQIFBgkKCwYGB/4+AaIFBAQEAWICAgIDBAQ
EBQUFBAMDawEBawMDBAUFCQgJCAcHBwYGBQQEAWIBAQBEBAGMEBAUGBgCHBwgJCAkJCQgICAcGBwU
FBQMDAgICAgMDBQUFBwYHCAgICQkODQ0MDAsLCQkIBwYEAWIBAwMEBgcICAoLCwwMDQ0ODg0NDQw
LCgoJBwcGBAQCAQECAwUGBwcJCgoLDA0NDew2BQUICikkkRIIERILCTcKGBQTEhwwHA8MDAUGOBM
4AwEBAQI4EzcLCwwRHTEcDRETEw0JOakUEBAUKSQpBwgGBQI2fHEt/JQCkC39QwYGBgsKCQYFAGe
BAGUGCQoLBgYGA2wGBgYLCgkGBQIBAAACAAAAAPza0wAAwALAAA3IRMhAzMTITUhJyFSAuq4/QP
rDrgCaf4uOv7dtAG9/kMB8Sh/AAAAAEAAAAAAxcD9AAFAAATCQEXCQHpaCn+NzMB+/4FA8H+P/4
/MwH0AfQAAAAAAQAAAAAD9A0AAAUAAAEnBwkBJwFZ52YBTQKbZwFM52b+sgKbZwAAAAIAAAAAA/Q
DngAIAA4AABMRmZUhFTMRJQUVCQE1AYzuAQnx/pL+BgH6Ae7+EgHT/o/09AFx84NwAVv+rneBUQA
AABIA3gABAAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACAABAAAAAADAACADwABAAA
AAAAEAACAFgABAAAAAAFAAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwA
DAAEECQAAAAIABQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4
AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyB1LWJjb25
zUmVndWxhcmUtYmNvbN1LWJjb25zVmVyc2lvbiAxLjB1LWJjb25zRm9udCBnZW51cmF0ZWQgdXN
pbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AYgBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAAtAGIAYwBvAG4AcwBlAC0AYgBjAG8AbgBzAFYAZQBByAHMAAQB
```

```
vAG4AIAAxAC4AMABlAC0AYgBjAG8AbgBzAEYAbwBuAHQAIABnAGUAbgBlAHIAyQB0AGUAZAAGAHU
AcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB
3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAYBAgEDAQQBBQEGAQcAE2RvY3VtZW50LXNldHRpbmctd2YOZm9sZGVyLW9
wZW4tMDERY2hldnJvbilyaWdodF8wMy0KY2hlY2stbWVfYawhob3VzZS0wNAAAAA=)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-icons {
    font-family: 'e-icons' !important;
    font-size: 14px;
}
</style>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

Cart ➤ Billing ➤ Shipping ➤ Payment

Customize Specific Item Template

The specific breadcrumb item can be customizable using itemTemplate with conditional rendering. In the following example, added the span element only for the **breadcrumb** text in breadcrumb item.

CSHTML

```
@Html.EJS().Breadcrumb("specific-item-template").CssClass("e-specific-item-template").ItemTemplate("#specificItemTemplate").Items(item =>
{
    item.Text("Home").Url("https://ej2.syncfusion.com/home/aspnetmvc.html").Add();

    item.Text("Components").Url("https://ej2.syncfusion.com/aspnetmvc/Grid/GridOverview").Add();

    item.Text("Navigations").Url("https://ej2.syncfusion.com/aspnetmvc/Menu/DefaultFunctionalities").Add();
    item.Text("Breadcrumb").Url("./DefaultFunctionalities").Add();
}).Render()
<script id="specificItemTemplate" type="text/x-template">
    <div>
        ${if(text=="Breadcrumb")}
        <span class="e-searchfor-text">
            <span style="margin-right: 5px">Search for:</span>
```

```

        <a class="e-breadcrumb-text" href="{url}" onclick="return
false">${text}</a>
    </span>
    ${else}
    <a class="e-breadcrumb-text" href="{url}" onclick="return
false">${text}</a>
    ${/if}
</div>
</script>
<script>
function beforeItemRenderHandler(args) {
    if (args.item.text !== 'Program Files') {
        args.item.disabled = true;
    }
}
</script>
@section Scripts {
    <script>
        document.getElementById("reset").addEventListener('click', function
() {
            var breadcrumb, breadcrumbInst, breadcrumbs =
document.querySelector('.content-wrapper').getElementsByClassName("e-
breadcrumb");
            for (var i = 0; i < breadcrumbs.length; i++) {
                breadcrumb = breadcrumbs[i];
                breadcrumbInst = ej.base.getComponent(breadcrumb,
'breadcrumb');
                breadcrumbInst.activeItem =
breadcrumbInst.items[breadcrumbInst.items.length - 1].text;
            }
        });
    </script>
<style>
    .e-searchfor-text {
        display: flex;
        align-items: center;
        font-size: 14px;
        font-weight: normal;
    }

    .e-searchfor-text .e-breadcrumb-text {
        padding-left: 0;
    }
    .e-bigger .e-searchfor-text {
        font-size: 16px
    }
    .fabric .e-searchfor-text,
    .fabric-dark .e-searchfor-text,
    .highcontrast .e-searchfor-text {
        font-size: 18px;
    }
    .e-bigger.fabric .e-searchfor-text,
    .e-bigger.fabric-dark .e-searchfor-text,
    .e-bigger.highcontrast .e-searchfor-text {
        font-size: 21px;
    }
    .e-specific-item-template .e-breadcrumb-item:last-child a:hover {

```



```
        text-decoration: underline;
    }
</style>
}
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

[Home](#) / [Components](#) / [Navigations](#) / Search for: [Breadcrumb](#)

Accessibility in ASP.NET MVC Breadcrumb control

The Breadcrumb component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Breadcrumb component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The Breadcrumb component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Breadcrumb component:

| Attributes | Purpose |

| --- | --- |

| **aria-label** | Indicates the breadcrumb item text. |

| **aria-disabled** | Indicates the state of breadcrumb item whether it is disabled. |

Keyboard interaction

The Breadcrumb component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Breadcrumb component.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Navigate to the next item and also next item in the popup of menu type overflow. |

| **Shift + Tab** | Navigate to the previous item also previous item in the popup of menu type overflow. |

| **Enter key in normal mode** | Select the breadcrumb item. |

| **Enter key in normal mode** | To open the popup of menu type overflow mode when you press enter on collapsed button and It will expand the items of collapsed type overflow mode when you press enter on collapsed button. |

Ensuring accessibility

The Breadcrumb component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Breadcrumb component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Breadcrumb component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Bullet Chart

Getting Started with ASP.NET MVC Bullet Chart Control

This section briefly explains about how to include [ASP.NET MVC Bullet Chart](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add script resources

Here, the script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Bullet Chart control

Now, add the Syncfusion ASP.NET MVC BulletChart control in `~/Home/Index.cshtml` page.

CSHTML

```
@(Html.EJS().BulletChart("container").Render())
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Bullet Chart control will be rendered in the default web browser.



Bullet Chart With Data

This section explains how to plot local data to the Bullet Chart.

The **value** and **target** values should be mapped with [ValueField](#) and [TargetField](#) respectively.

CSHTML

```
@model List<BulletChartSample.Controllers.BulletChartData>
@ (Html.EJS().BulletChart("container")
    .ValueField("value")
```

```
.TargetField("target")
.Minimum(0).Maximum(300).Interval(50)
.DataSource(ViewBag.dataSource)
.Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<BulletChartData> data = new List<BulletChartData>
    {
        new BulletChartData { value = 270, target = 250 }
    };
    return View(data);
}

public class BulletChartData
{
    public double target;
    public double value;
}
```

Add Bullet Chart Title

You can add a title using [Title](#) property to the Bullet Chart to provide quick information to the user about the data plotted in the Bullet Chart.

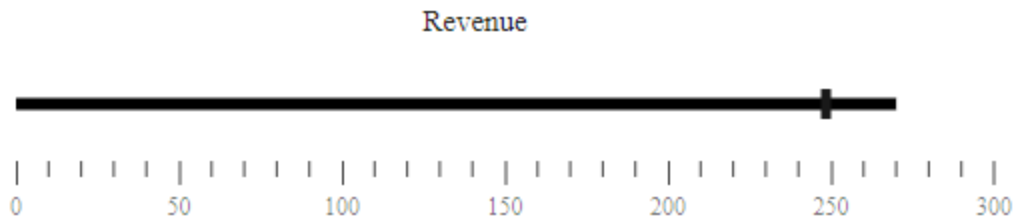
CSHTML

```
@model List<BulletChartSample.Controllers.BulletChartData>
@ (Html.EJS().BulletChart("container")
    .Title("Revenue")
    .ValueField("value")
    .TargetField("target")
    .Minimum(0).Maximum(300).Interval(50)
    .DataSource(ViewBag.dataSource)
    .Render())
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<BulletChartData> data = new List<BulletChartData>
    {
        new BulletChartData { value = 270, target = 250 }
    };
    return View(data);
}

public class BulletChartData
{
    public double target;
    public double value;
}
```



Ranges

You can add a range using [Ranges](#) property to the Bullet Chart.

CSHTML

```
@model List<BulletChartSample.Controllers.BulletChartData>
@(Html.EJS().BulletChart("container")
    .Title("Revenue")
    .ValueField("value")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(100).Add();
        rn.End(200).Add();
        rn.End(300).Add();
    })
    .Minimum(0).Maximum(300).Interval(50)
    .DataSource(Model)
    .Render())
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<BulletChartData> data = new List<BulletChartData>
    {
        new BulletChartData { value = 270, target = 250 }
    };
    return View(data);
}

public class BulletChartData
{
    public double target;
    public double value;
}
```

Tooltip

You can use tooltip for the Bullet Chart by setting the [Enable](#) property to true in **Tooltip**.

CSHTML

```
@model List<BulletChartSample.Controllers.BulletChartData>
@(Html.EJS().BulletChart("container")
    .Title("Revenue")
    .Tooltip(tp => tp.Enable(true))
```

```

.ValueField("value")
.TargetField("target")
.Ranges(rn =>
{
    rn.End(150).Add();
    rn.End(250).Add();
    rn.End(300).Add();
})
.Minimum(0).Maximum(300).Interval(50)
.DataSource(Model)
.Render()

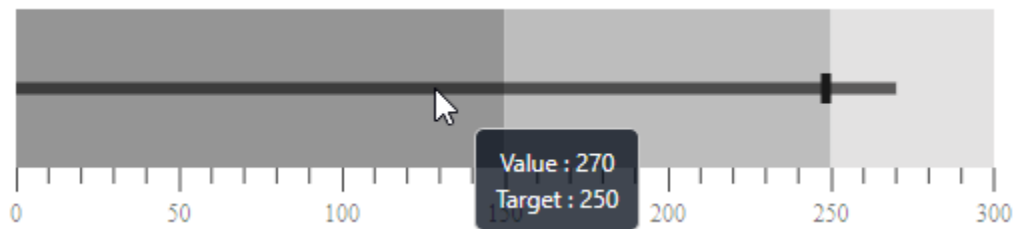
```

HOMECONTROLLER.CS

```

public ActionResult Index()
{
    List<BulletChartData> data = new List<BulletChartData>
    {
        new BulletChartData { value = 270, target = 250 }
    };
    return View(data);
}
public class BulletChartData
{
    public double target;
    public double value;
}

```



Note: [View Sample in GitHub.](#)

Bullet chart dimensions

Size for container

The size of the Bullet Chart is determined by the container size, and it can be changed inline or via CSS as following.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Revenue")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
    .Width("600")
    .Height("100")
    .Ranges(rn =>

```

```

        {
            rn.End(100).Add();
            rn.End(200).Add();
            rn.End(300).Add();
        })
        .Minimum(0).Maximum(300).Interval(50)
        .DataSource(ViewBag.dataSource)
        .Render()
    }
}

```

CONTAINER.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 270, target = 250 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```

Size for bullet chart

The [Width](#) and [Height](#) properties are used to adjust the size of the Bullet Chart.

Pixel

Sets the size of the Bullet Chart in pixels as shown below.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Revenue")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
    .Width("600px")
    .Height("100px")
    .Ranges(rn =>
    {
        rn.End(100).Add();
        rn.End(200).Add();
        rn.End(300).Add();
    })
    .Minimum(0).Maximum(300).Interval(50)
    .DataSource(ViewBag.dataSource)
    .Render()

```

PIXEL.CS

```

public IActionResult Index()
{

```



```

        List<DefaultBulletData> bulletData = new List<DefaultBulletData>
        {
            new DefaultBulletData { value = 270, target = 250 }
        };
        ViewBag.dataSource = bulletData;
        return View();
    }

    public class DefaultBulletData
    {
        public double value;
        public double target;
    }

```

Percentage

By setting a value in percentage, the Bullet Chart gets its dimension with respect to its container. For example, when the height is **50%**, the Bullet Chart renders to half of the container's height.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Revenue")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
    .Width("80%")
    .Height("90%")
    .Ranges(rn =>
    {
        rn.End(100).Add();
        rn.End(200).Add();
        rn.End(300).Add();
    })
    .Minimum(0).Maximum(300).Interval(50)
    .DataSource(ViewBag.dataSource)
    .Render()

```

PERCENTAGE.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 270, target = 250 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```

Note: If the size is not specified, the Bullet Chart will be rendered with a height of **126px** and a width of the window.

Axis customization

MajorTickLines and MinorTickLines customization

You can customize the [Width](#), [Color](#), and [Height](#) of minor and major tick lines using the [MajorTickLines](#) and [MinorTickLines](#) properties of the bullet chart.

The following properties can be used to customize **MajorTicklines** and **MinorTicklines**.

- **Width** - Specifies the width of ticklines.
- **Height** - Specifies the height of ticklines.
- **Color** - Specifies the color of ticklines.
- **UseRangeColor** - Specifies the color of ticklines and represents the color from corresponding range colors.

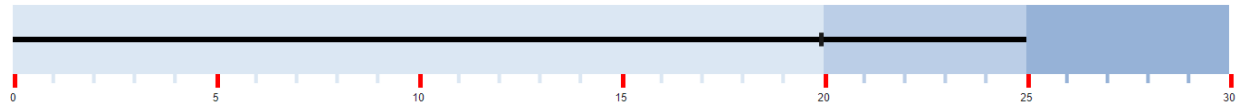
CSHTML

```
@(Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .Tooltip(tp => tp.Enable(true))
    .MajorTickLines(ml => ml.Color("red").Width(5))
    .MinorTickLines(mi => mi.Width(4).Color("blue"))
    .ValueField("value")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(500).Add();
        rn.End(1500).Add();
        rn.End(2500).Add();
    })
    .Minimum(0).Maximum(2500).Interval(250)
    .DataSource(ViewBag.dataSource)
    .Render()
)
```

TICKS.CS

```
public ActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```



Tick placement

The major and the minor ticks can be placed **Inside** or **Outside** the ranges using the [TickPosition](#) property.

CSHTML

```
@(Html.EJS().BulletChart("container")
    .Tooltip(tp => tp.Enable(true))

    .TickPosition(Syncfusion.EJ2.Charts.TickPosition.Inside)
    .ValueField("value")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(500).Add();
        rn.End(1500).Add();
        rn.End(2500).Add();
    })
    .Minimum(0).Maximum(2500).Interval(250)
    .DataSource(ViewBag.dataSource)
    .Render()
)
```

TICK-PLACEMENT.CS

```
public ActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```

Label format

Axis numeric labels can be formatted by using the [LabelFormat](#) property. Axis labels support all globalize formats.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
```

```

.Ranges(rn =>
{
    rn.End(500).Add();
    rn.End(1500).Add();
    rn.End(2500).Add();
})
.Minimum(0).Maximum(2500).Interval(250)
.DataSource(ViewBag.dataSource)
.Render()

```

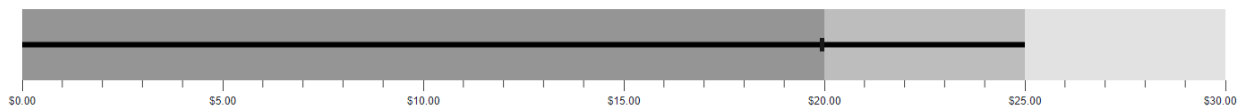
LABEL-FORMAT.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



The following table describes the result of applying some commonly used formats to numeric axis labels.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal places
1000	n3	1000.000	The Number is rounded to 3 decimal places
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal places
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal places
1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place

1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal places
------	----	-----------	---

Grouping separator

To separate groups of thousands, use the [EnableGroupSeparator](#) property of bullet-chart. To separate the groups of thousands, set the [EnableGroupSeparator](#) property to **true**.

CSHTML

```
@(Html.EJS().BulletChart("container")
    .ValueField("value")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(500).Add();
        rn.End(1500).Add();
        rn.End(2500).Add();
    })
    .Minimum(0).Maximum(2500).Interval(250)
    .DataSource(ViewBag.dataSource)
    .Render()
)
```

GROUPING.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```

Custom label format

Using the [LabelFormat](#) property, axis labels can be specified with a custom defined format in addition to the axis value. The label format uses a placeholder such as **`\${value}K**, which represents the axis label.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
    .LabelFormat("${value}K")
    .Ranges(rn =>
```

```

        {
            rn.End(500).Add();
            rn.End(1500).Add();
            rn.End(2500).Add();
        })
        .Minimum(0).Maximum(2500).Interval(250)
        .DataSource(ViewBag.dataSource)
        .Render()
    }
}

```

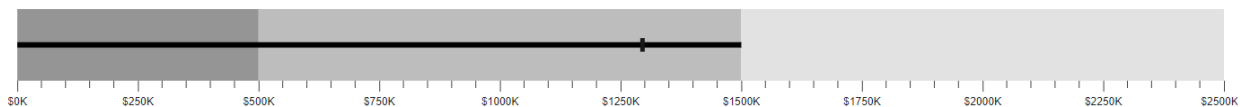
CUSTOM-LABEL.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Label placement

You can customize the axis labels **Inside** or **Outside** the bullet-chart using the [LabelPosition](#) property.

CSHTML

```

@ (Html.EJS().BulletChart("container")

.LabelPosition(Syncfusion.EJ2.Charts.LabelsPlacement.Inside)
    .ValueField("value")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(20).Add();
        rn.End(25).Add();
        rn.End(30).Add();
    })
    .Minimum(0).Maximum(30).Interval(5)
    .DataSource(ViewBag.dataSource)
    .Render()

)

```

LABEL-PLACEMENT.CS

```

public IActionResult Index()
{

```

```

        List<DefaultBulletData> bulletData = new List<DefaultBulletData>
        {
            new DefaultBulletData { value = 1500, target = 1300 }
        };
        ViewBag.dataSource = bulletData;
        return View();
    }

    public class DefaultBulletData
    {
        public double value;
        public double target;
    }

```

Opposed position

To place an axis opposite to its original position, set the [OpposedPosition](#) property to **true**.

CSHTML

```

@ (Html.EJS().BulletChart("container")
    .OpposedPosition(true)
    .ValueField("value")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(500).Add();
        rn.End(1500).Add();
        rn.End(2500).Add();
    })
    .Minimum(0).Maximum(2500).Interval(250)
    .DataSource(ViewBag.dataSource)
    .Render()
)

```

OPPOSED.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```

Category label

The Bullet Chart supports X-axis label by specifying the property from the data source to the [CategoryField](#). It helps to understand the input data in a more efficient way.

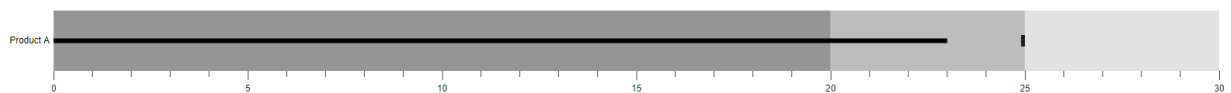
CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
    .CategoryField("category")
    .Ranges(rn =>
    {
        rn.End(500).Add();
        rn.End(1500).Add();
        rn.End(2500).Add();
    })
    .Minimum(0).Maximum(2500).Interval(250)
    .DataSource(ViewBag.dataSource)
    .Render();
```

CATEGORY.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300,
category = "Product A" }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
    public string category;
}
```

**Category label customization**

The label color, opacity, font size, font family, font weight, and font style can be customized by using the [CategoryLabelStyle](#) setting for category and the [LabelStyle](#) setting for axis label. The [UseRangeColor](#) property specifies the color of the axis label and represents the color from the corresponding range colors.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("target")
    .CategoryField("category")
```



```

.Ranges(rn =>
{
    rn.End(500).Add();
    rn.End(1500).Add();
    rn.End(2500).Add();
})
.Minimum(0).Maximum(2500).Interval(250)
.DataSource(ViewBag.dataSource)
.Render()

```

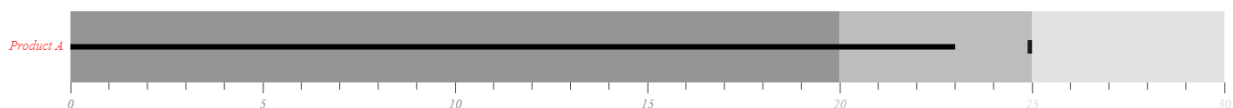
CATEGORY-LABEL.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 1500, target = 1300,
category = "Product A" }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
    public string category;
}

```



Working with data

Bullet Chart can visualise data bound from local or remote data.

Local data

You can bind a simple JSON data to the chart using [DataSource](#) direct property of the bullet-chart. Now, map the fields in JSON to [ValueField](#) and [TargetField](#) properties. The [DataSource](#) property accepts a collection of values as input that helps to display measures, and compares them to a target bar. To display the actual and target bar, specify the property from the datasource into the [ValueField](#) and [TargetField](#) respectively.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Profit in %")
    .ValueField("value")
    .TargetField("comparativeMeasureValue")
    .CategoryField("category")
    .Ranges(rn =>
    {
        rn.End(5).Add();
        rn.End(15).Add();
    }

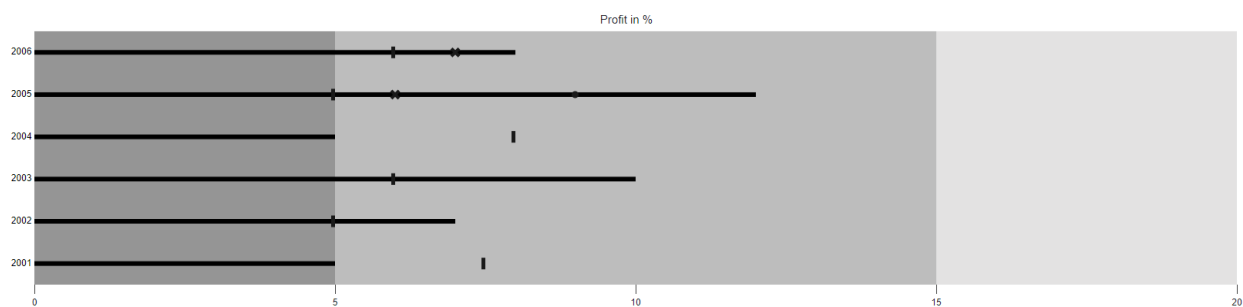
```

```
rn.End(20).Add();
    })
    .Minimum(0).Maximum(20).Interval(5)
    .DataSource(ViewBag.dataSource)
    .Render();
```

WORKING-WITH-DATA.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 5, comparativeMeasureValue = 7.5, category= "2001"},
        new DefaultBulletData { value = 7, comparativeMeasureValue = 5, category= "2002"},
        new DefaultBulletData { value = 10, comparativeMeasureValue = 6, category= "2003"},
        new DefaultBulletData { value = 5, comparativeMeasureValue = 8, category= "2004"},
        new DefaultBulletData { value = 12, comparativeMeasureValue = 5, category= "2005"},
        new DefaultBulletData { value = 8, comparativeMeasureValue = 6, category= "2006"}
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double comparativeMeasureValue;
    public string category;
}
```



Ranges

Ranges represent the quality of a specific range such as **Good**, **Bad** and **Satisfactory** in the Bullet Chart scale. The ending point of a qualitative range is specified in the [End](#) property in [Ranges](#). The [Minimum](#) value of a quantitative scale is considered the starting point of the first range or the previous range end point.

CSHTML

```
@Html.EJS().BulletChart("container")
```

```

        .Title("Sales Rate")
        .ValueField("value")
        .TargetField("target")
        .CategoryField("category")
        .CategoryLabelStyle(cl =>
        {
            cl.Color("red").Size("13").FontWeight("Bold").Add();
        })
        .Ranges(rn =>
        {
            rn.End(35).Add();
            rn.End(50).Add();
            rn.End(100).Add();
        })
        .Minimum(0).Maximum(100).Interval(20)
        .DataSource(ViewBag.dataSource)
        .Render()

```

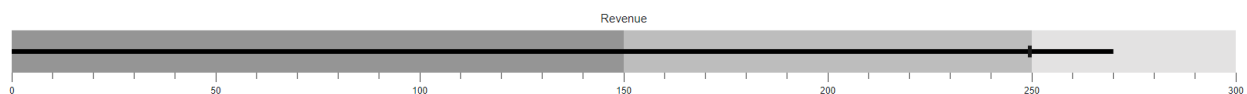
RANGES.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75, category =
"Year 1"},
        new DefaultBulletData { value = 70, target = 70, category =
"Year 2"},
        new DefaultBulletData { value = 85, target = 75, category =
"Year 3"}
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
    public string category;
}

```



Color customization

Enhance the readability of ranges with color and opacity. It can be applied using the [Color](#) and [Opacity](#) properties in [Ranges](#).

CSHTML

```

@Html.EJS().BulletChart("container")
        .Title("Sales Rate")
        .ValueField("value")
        .TargetField("target")

```

```

        .CategoryField("category")
        .CategoryLabelStyle(cl =>
        {
cl.Color("red").Size("13").FontWeight("Bold").Add();
        })
        .Ranges(rn =>
        {
            rn.End(35).Opacity(0.5).Color("darkred").Add();
            rn.End(50).Opacity(1).Color("red").Add();
            rn.End(75).Opacity(0.7).Color("blue").Add();
            rn.End(90).Opacity(1).Color("lightgreen").Add();
            rn.End(100).Opacity(0.8).Color("green").Add();
        })
        .Minimum(0).Maximum(100).Interval(20)
        .DataSource(ViewBag.dataSource)
        .Render()

```

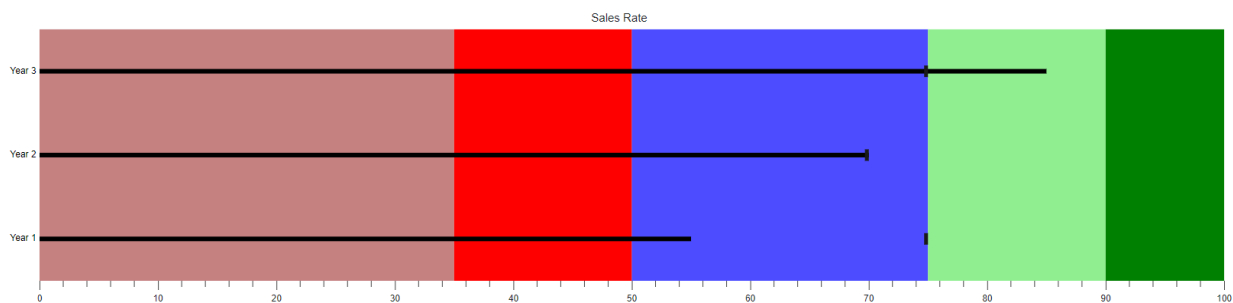
RANGES-CUSTOM.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75, category =
"Year 1"},
        new DefaultBulletData { value = 70, target = 70, category =
"Year 2"},
        new DefaultBulletData { value = 85, target = 75, category =
"Year 3"}
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
    public string category;
}

```



Actual bar

To display the primary data or the current value of the data being measured known as the **Feature Measure** that should be encoded as a bar. This is called as the **Actual Bar** or the **Feature Bar** in the

Bullet Chart, and to display the actual bar the [ValueField](#) should be mapped to the appropriate field from the data source.

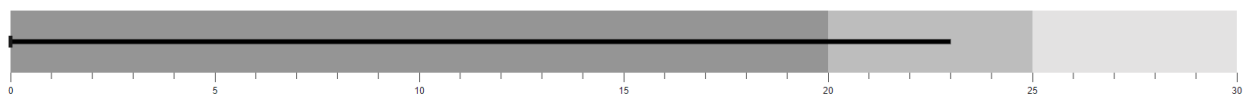
CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .ValueField("value")
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render();
```

VALUE-BAR.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```



Types of actual bar

The shape of the actual bar can be customized using the [Type](#) property of the Bullet Chart. The actual bar contains **Rect** and **Dot** shapes. By default, the actual bar shape is Rect.

CSHTML

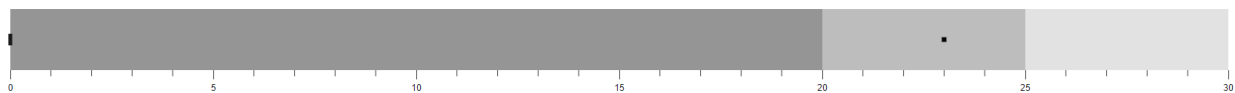
```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .ValueField("value")
    .Type("Dot")
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
```

```
.DataSource(ViewBag.dataSource)
.Render()
```

TYPES.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```



Actual bar customization

Border customization

Using the [ValueBorder](#) property of the bullet chart, you can customize the border [Color](#) and [Width](#) of the actual bar.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .ValueField("value")
    .ValueBorder(vb =>
    {
        vb.Color("red").Width(3).Add();
    })
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()
```

VALUE-BORDER.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
}
```

```

    };
    ViewBag.dataSource = bulletData;
    return View();
}
public class DefaultBulletData
{
    public double value;
    public double target;
}

```

Fill color and height customization

Customize the fill color and height of the actual bar using the [ValueFill](#) and [ValueHeight](#) properties of the bullet chart. Also, you can bind the color for the actual bar from [DataSource](#) for the bullet chart using [ValueFill](#) property.

CSHTML

```

@{
    var color = "blue";
}
@(Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .ValueField("value")
    .ValueFill(@color)
    .ValueHeight(15)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render())

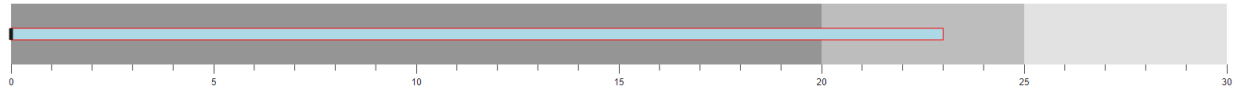
```

VALUE-FILL.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}
public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Target bar

The line marker that runs perpendicular to the orientation of the graph is known as the **Comparative Measure** and it is used as a target marker to compare against the feature measure value. This is also called as the **Target Bar** in the Bullet Chart. To display the target bar, the [TargetField](#) should be mapped to the appropriate field from the datasource.

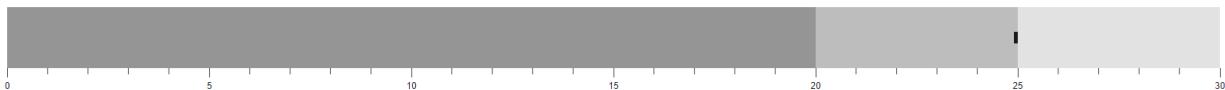
CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .TargetField("target")
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render();
```

TARGET-BAR.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```



Types of target bar

The shape of the target bar can be customized using the [TargetTypes](#) property and it supports **Circle**, **Cross**, and **Rect** shapes. The default type of the target bar is **Rect**.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .TargetField("target")
```



```

.TargetTypes(["Circle"])
.Ranges(rn =>
{
    rn.End(35).Add();
    rn.End(50).Add();
    rn.End(100).Add();
})
.Minimum(0).Maximum(100).Interval(20)
.DataSource(ViewBag.dataSource)
.Render()

```

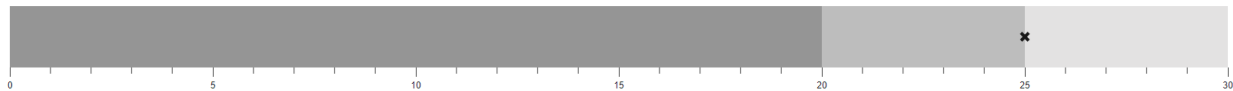
TARGET-TYPES.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Target bar customization

The following properties can be used to customize the target bar. Also, you can bind the color for the target bar from [DataSource](#) for the bullet chart.

- [TargetColor](#) - Specifies the fill color of target bar.
- [TargetWidth](#) - Specifies the width of target bar.

CSHTML

```

@{
    var color = "red";
}
@(Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .TargetField("target")
    .TargetColor(@color)
    .TargetWidth(15)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    }

```

```

    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

```

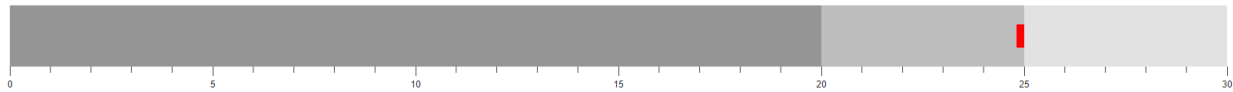
TARGET-COLOR.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Title and subtitle

Title

The title of the Bullet Chart displays the information about the data plotted by specifying it in the [Title](#) property.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Sales Rate")
    .TargetField("target")
    .ValueField("value")
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

```

TITLE.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    }
}

```

```

    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Subtitle

To show additional information about the data plotted, the Bullet Chart can also be given a subtitle using the [Subtitle](#) property.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

```

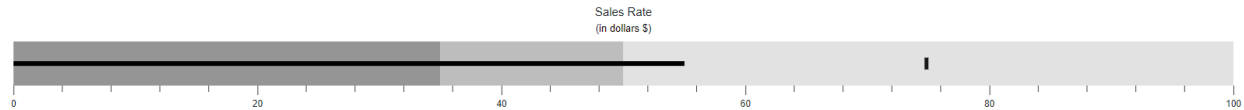
SUB-TITLE.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Title and subTitle position

The title and the subtitle positions can be customized using the [TitlePosition](#) property. Possible positions are **Left**, **Right**, **Top**, and **Bottom**.

Position as left

By setting the [TitlePosition](#) to **Left**, you can display the title and subtitle at the left side of the Bullet Chart.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")

    .TitlePosition(Syncfusion.EJ2.Charts.TextPosition.Left)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()
```

LEFT.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```

Position as right

By setting the [TitlePosition](#) to **Right**, you can display the title and subtitle at the right side of the Bullet Chart.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")

.TitlePosition(Syncfusion.EJ2.Charts.TextPosition.Right)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()
```

RIGHT.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```

Position as top

By setting the [TitlePosition](#) to **Top**, you can display the title and subtitle at the top of the Bullet Chart. The default title and subtitle positions of the Bullet Chart is **Top**.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")

.TitlePosition(Syncfusion.EJ2.Charts.TextPosition.Top)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
```

```
.Minimum(0).Maximum(100).Interval(20)
.DataSource(ViewBag.dataSource)
.Render()
```

TOP.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```

Position as bottom

By setting the [TitlePosition](#) to **Bottom**, you can display the title and subtitle at the bottom of the Bullet Chart.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")

.TitlePosition(Syncfusion.EJ2.Charts.TextPosition.Bottom)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()
```

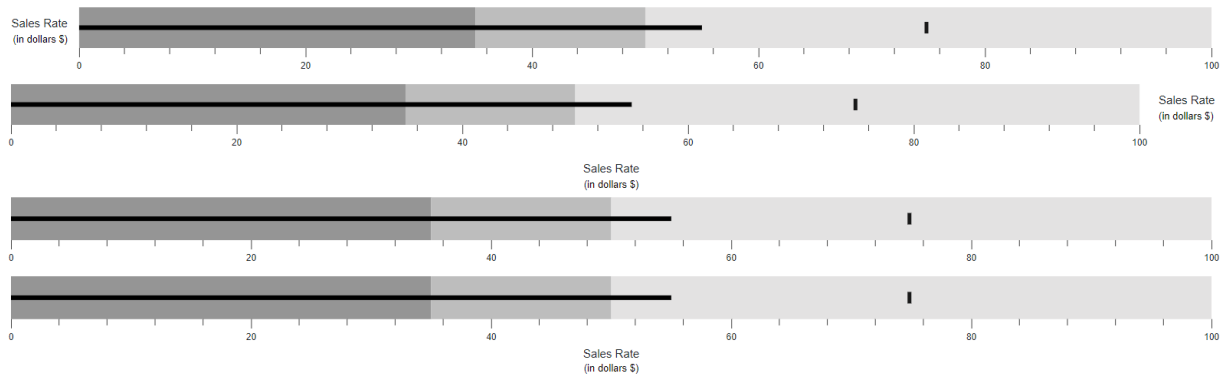
BOTTOM.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}
```

```

    }
    public class DefaultBulletData
    {
        public double value;
        public double target;
    }

```



Title customization

The title color, opacity, font size, font family, font weight, and font style can be customized using the [TitleStyle](#) property.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .TitleStyle(ts =>
    {
        ts.Size("22px").Color("red").FontFamily("cursive").FontWeight("Bold").Add();
    })
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

```

TITLE-CUSTOM.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
}

```

```

        ViewBag.dataSource = bulletData;
        return View();
    }
    public class DefaultBulletData
    {
        public double value;
        public double target;
    }

```

SubTitle customization

The sub-title color, opacity, font size, font family, font weight, and font style can be customized using the [SubtitleStyle](#) property.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .SubtitleStyle(ts =>
    {
        ts.Size("22px").Color("red").FontFamily("cursive").FontWeight("Bold").Add();
    })
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

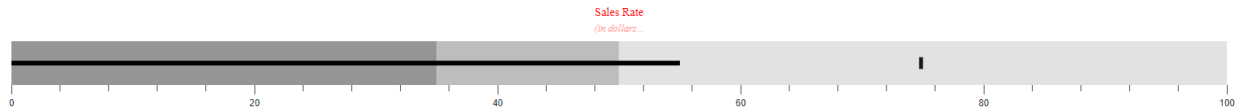
```

SUB-TITLE-CUSTOM.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}
public class DefaultBulletData
{
    public double value;
    public double target;
}

```

Customization

Orientation

The Bullet Chart can be rendered in different orientations such as **Horizontal** or **Vertical** via the [Orientation](#) property. By default, the Bullet Chart is rendered in the **Horizontal** orientation.

CSHTML

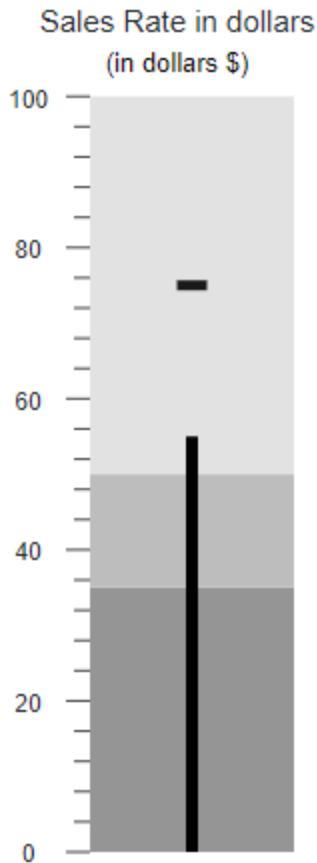
```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .Width("20%")

.Orientation(Syncfusion.EJ2.Charts.OrientationType.Vertical)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()
```

ORIENTATION.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}
```



Right-to-left (RTL)

The Bullet Chart supports the right-to-left rendering that can be enabled by setting the [EnableRtl](#) property to **true**.

CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .EnableRtl(true)
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()
```

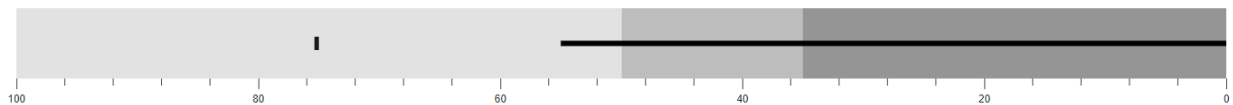
RIGHT-TO-LEFT.CS

```
public IActionResult Index()
```

```

    {
        List<DefaultBulletData> bulletData = new List<DefaultBulletData>
        {
            new DefaultBulletData { value = 55, target = 75 }
        };
        ViewBag.dataSource = bulletData;
        return View();
    }
    public class DefaultBulletData
    {
        public double value;
        public double target;
    }

```



Animation

The actual and the target bar supports the linear animation via the [Animation](#) setting. The speed and the delay are controlled using the [Duration](#) and [Delay](#) properties respectively.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .Animation(an => an.Enable(true))
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

```

ANIMATION.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}
public class DefaultBulletData
{
    public double value;

```

```

        public double target;
    }

```

Theme

The Bullet Chart supports different type of themes via the [Theme](#) property.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Sales Rate in dollars")
    .Subtitle("(in dollars $)")
    .LabelFormat("${value}")
    .TargetField("target")
    .ValueField("value")
    .Theme("HighContrast")
    .Ranges(rn =>
    {
        rn.End(35).Add();
        rn.End(50).Add();
        rn.End(100).Add();
    })
    .Minimum(0).Maximum(100).Interval(20)
    .DataSource(ViewBag.dataSource)
    .Render()

```

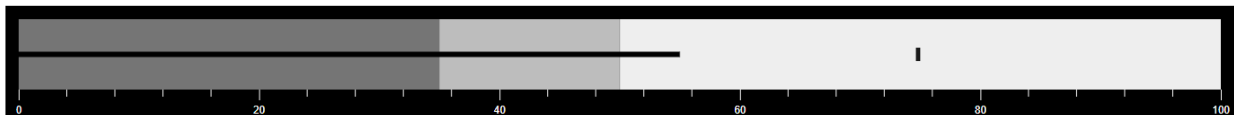
THEME.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 55, target = 75 }
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double target;
}

```



Data label

Data Labels are used to identify the value of actual bar in the Bullet Chart component. The Data Labels will be shown by specifying the [DataLabel](#) setting's [Enable](#) property to **true**.

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Profit in percentage")

```

```

        .ValueField("value")
        .TargetField("comparativeMeasureValue")
        .CategoryField("category")
        .Height("400")
        .LabelFormat("{value}%")
        .DataLabel(db => db.Enable(true))
        .Ranges(rn =>
        {
            rn.End(5).Add();
            rn.End(15).Add();
            rn.End(20).Add();
        })
        .Minimum(0).Maximum(20).Interval(5)
        .DataSource(ViewBag.dataSource)
        .Render()

```

DATA-LABEL.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 5, comparativeMeasureValue =
7.5, category= "2001"},
        new DefaultBulletData { value = 7, comparativeMeasureValue =
5, category= "2002"},
        new DefaultBulletData { value = 10, comparativeMeasureValue
= 6, category= "2003"},
        new DefaultBulletData { value = 5, comparativeMeasureValue =
8, category= "2004"},
        new DefaultBulletData { value = 12, comparativeMeasureValue
= 5, category= "2005"},
        new DefaultBulletData { value = 8, comparativeMeasureValue =
6, category= "2006"}
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double comparativeMeasureValue;
    public string category;
}

```

Data label customization

Data Labels color, opacity, font size, font family, font weight, and font style can be customized using the [LabelStyle](#).

CSHTML

```

@Html.EJS().BulletChart("container")
    .Title("Profit in percentage")
    .ValueField("value")
    .TargetField("comparativeMeasureValue")

```

```

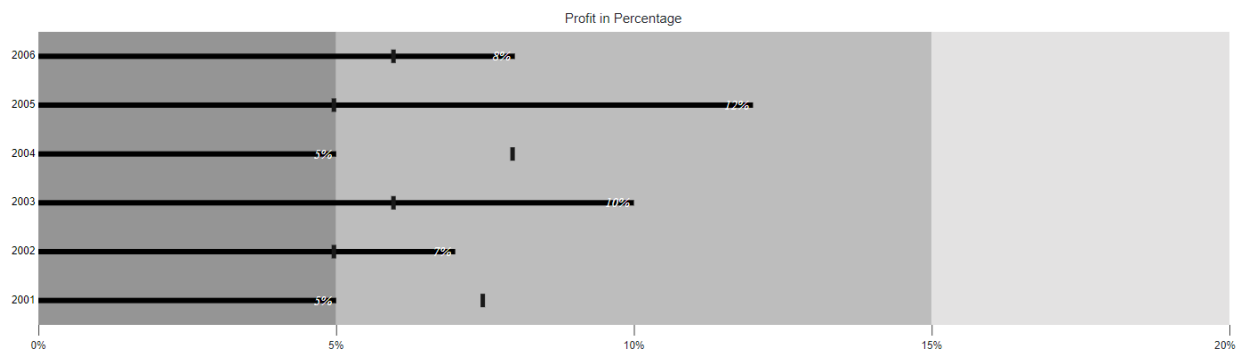
        .CategoryField("category")
        .Height("400")
        .LabelFormat("{value}%")
        .DataLabel(db => db.Enable(true))
        .Ranges(rn =>
        {
            rn.End(5).Add();
            rn.End(15).Add();
            rn.End(20).Add();
        })
        .Minimum(0).Maximum(20).Interval(5)
        .DataSource(ViewBag.dataSource)
        .Render()
    
```

DATA-LABEL-CUSTOM.CS

```

public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 5, comparativeMeasureValue =
7.5, category= "2001"},
        new DefaultBulletData { value = 7, comparativeMeasureValue =
5, category= "2002"},
        new DefaultBulletData { value = 10, comparativeMeasureValue
= 6, category= "2003"},
        new DefaultBulletData { value = 5, comparativeMeasureValue =
8, category= "2004"},
        new DefaultBulletData { value = 12, comparativeMeasureValue
= 5, category= "2005"},
        new DefaultBulletData { value = 8, comparativeMeasureValue =
6, category= "2006"}
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double comparativeMeasureValue;
    public string category;
}
    
```



Tooltip

When the mouse is hovered over a bar in the Bullet Chart, the tooltip displays important summary about the actual and the target bar values.

Default tooltip

By setting [Enable](#) property to true. By default tooltip is visible in bullet-chart. The tooltip is not visible by default. To make it visible, set the [Enable](#) property in the [Tooltip](#) to **true**.

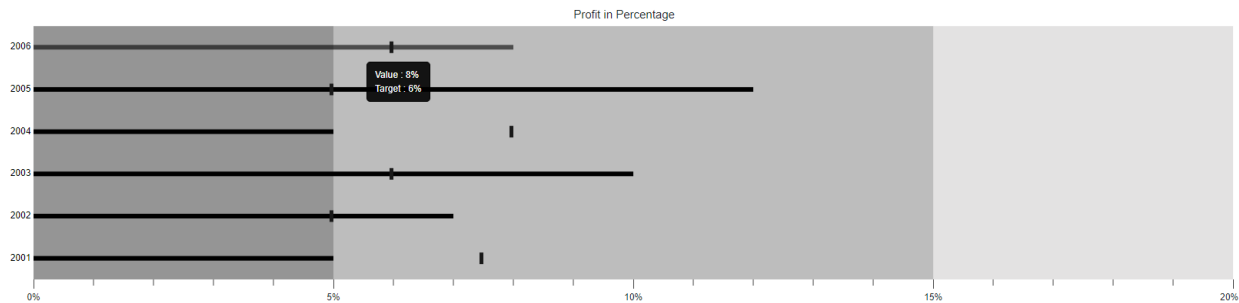
CSHTML

```
@Html.EJS().BulletChart("container")
    .Title("Profit in percentage")
    .Tooltip(tp => tp.Enable(true))
    .ValueField("value")
    .TargetField("comparativeMeasureValue")
    .CategoryField("category")
    .Height("400")
    .LabelFormat("{value}%")
    .Ranges(rn =>
    {
        rn.End(5).Add();
        rn.End(15).Add();
        rn.End(20).Add();
    })
    .Minimum(0).Maximum(20).Interval(5)
    .DataSource(ViewBag.dataSource)
    .Render()
```

TOOL-TIP.CS

```
public IActionResult Index()
{
    List<DefaultBulletData> bulletData = new List<DefaultBulletData>
    {
        new DefaultBulletData { value = 5, comparativeMeasureValue =
7.5, category= "2001"},
        new DefaultBulletData { value = 7, comparativeMeasureValue =
5, category= "2002"},
        new DefaultBulletData { value = 10, comparativeMeasureValue
= 6, category= "2003"},
        new DefaultBulletData { value = 5, comparativeMeasureValue =
8, category= "2004"},
        new DefaultBulletData { value = 12, comparativeMeasureValue
= 5, category= "2005"},
        new DefaultBulletData { value = 8, comparativeMeasureValue =
6, category= "2006"}
    };
    ViewBag.dataSource = bulletData;
    return View();
}

public class DefaultBulletData
{
    public double value;
    public double comparativeMeasureValue;
    public string category;
}
```



Tooltip template

Any HTML elements can be displayed in the tooltip by using the [Template](#) property of the [Tooltip](#). You can use the `#{target}` and `#{value}` as place holders in the HTML element to display the value and target values from the data source of corresponding data point.

Tooltip customization

The following properties can be used to customize the Bullet Chart tooltip.

The [Fill](#) and [Border](#) properties are used to customize the background color and border of the tooltip respectively. The [TextStyle](#) property in the tooltip is used to customize the font of the tooltip text.

- **Fill** - Specifies the color of tooltip.
- **Border** - Specifies the tooltip border color and width.
- **TextStyle** - Specifies the tooltip font family, font style, font weight, color and size.

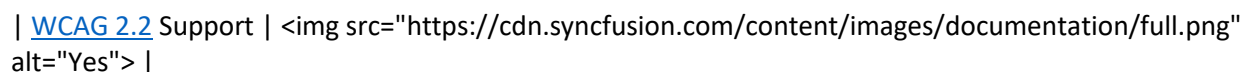
Accessibility in ASP.NET MVC Bullet chart component

The Bullet chart component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Bullet chart component is outlined below.

| Accessibility Criteria | Compatibility |

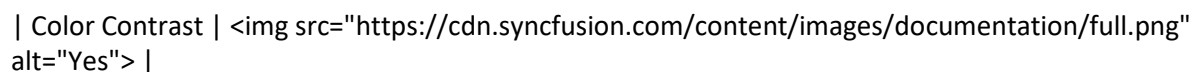
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |


```

| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The Bullet chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Bullet chart component:

- `img (role)`
- `button (role)`
- `aria-label (attribute)`
- `aria-pressed (attribute)`

Keyboard interaction

The Bullet chart component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Bullet chart component.

| Press | To do this |

| --- | --- |

| Tab | Moves the focus to the next element in the Bullet chart. |

| Shift + Tab | Moves the focus to the previous element in the Bullet chart. |

| Ctrl + P | Prints the Bullet chart. |

Ensuring accessibility

The Bullet chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Bullet chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Bullet chart component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Button Group

Getting Started

This section briefly explains about how to include a simple [ASP.NET MVC ButtonGroup](#) in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ButtonGroup to the project

To create simple ButtonGroup add the div tag with class name as `e-btn-group` and add `Button` component that should inside the `div` element in your `Index.cshtml` page.

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Output be like the below.

HTML CSS JAVASCRIPT

Run the application

After successful compilation of your application, simply press **F5** to run the application.

The following example shows a default rendering of ButtonGroup.

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Orientation

ButtonGroup can be arranged in a vertical and horizontal orientation. By default, it is horizontally aligned.

Vertical Orientation

ButtonGroup can be aligned vertically by using the built-in CSS class **e-vertical** to the target element.

The following example illustrates how to achieve vertical orientation in ButtonGroup.

CSHTML

```
<div class='e-btn-group e-vertical'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: ButtonGroup does not support SplitButton component nesting in a vertical orientation.

See Also

- [Initialize ButtonGroup using util function](#)

Types and Styles in Button Group Control

This section explains about different types and styles of ButtonGroup.

ButtonGroup types

Outline ButtonGroup

An Outline ButtonGroup has a border with transparent background. To create Outline ButtonGroup, **e-outline** class should be added to the target element and to each button elements using **CssClass** property.

CSHTML

```
<div class='e-btn-group e-outline'>
    @Html.EJS().Button("html").Content("HTML").CssClass("e-
outline").Render()
    @Html.EJS().Button("css").Content("CSS").CssClass("e-outline").Render()
    @Html.EJS().Button("javascript").Content("Javascript").CssClass("e-
outline").Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: ButtonGroup does not have support for flat and round types.

ButtonGroup styles

The Essential JS 2 ButtonGroup has the following predefined styles. This can be achieved by adding corresponding class name in each button elements using **CssClass** property.

Class	Description
-----	-----
e-primary	Used to represent a primary action.
e-success	Used to represent a positive action.
e-info	Used to represent an informative action.
e-warning	Used to represent an action with caution.
e-danger	Used to represent a negative action.

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("view").Content("View").CssClass("e-info").Render()
    @Html.EJS().Button("edit").Content("Edit").Render()
    @Html.EJS().Button("delete").Content("Delete").CssClass("e-
danger").Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: Predefined ButtonGroup styles provide only the visual indication. So, ButtonGroup content should define the ButtonGroup style for the users of assistive technologies such as screen readers.

Note: [View Sample in GitHub.](#)

See also

- [ButtonGroup with icons](#)
- [Create ButtonGroup with rounded corner](#)

Selection and Nesting

Selection

Single selection

ButtonGroup supports radio type selection in which only one button can be selected. This can be achieved by adding input element along with `id` attribute with its corresponding label along with `for` attribute inside the target element. In this ButtonGroup, the type of the input element should be `radio` and `e-btn` is added to the `label` element.

CSHTML

```
<div class='e-btn-group'>
    <input type="radio" id="radioleft" name="align" value="left" />
    <label class="e-btn" for="radioleft">Left</label>
    <input type="radio" id="radiomiddle" name="align" value="middle" />
    <label class="e-btn" for="radiomiddle">Center</label>
    <input type="radio" id="radiatoright" name="align" value="right" />
    <label class="e-btn" for="radiatoright">Right</label>
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Multiple selection

ButtonGroup supports checkbox type selection in which multiple button can be selected. This can be achieved by adding input element along with `id` attribute with its corresponding label along with `for` attribute inside the target element. In this ButtonGroup, the type of the input element should be `checkbox` and `e-btn` is added to the `label` element.

CSHTML

```
<div class='e-btn-group'>
    <input type="checkbox" id="checkbold" name="font" value="bold" />
    <label class="e-btn" for="checkbold">Bold</label>
</div>
```

```

<input type="checkbox" id="checkitalic" name="font" value="italic" />
<label class="e-btn" for="checkitalic">Italic</label>
<input type="checkbox" id="checkline" name="font" value="underline"/>
<label class="e-btn" for="checkline">Underline</label>
</div>

```

DEFAULT.CS

```

public ActionResult Index()
{
    return View();
}

```

Nesting

Nesting with other components can be possible in ButtonGroup. The following components can be nested in ButtonGroup.

- DropDownButton
- SplitButton

DropDownButton

To initialize DropDownButton component refer [DropDownButton Getting Started documentation](#).

CSHTML

```

<div class='e-btn-group'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()

    @Html.EJS().DropDownButton("element").Content("More").Items((IEnumerable<object>)ViewBag.items).Render()
</div>

```

DEFAULT.CS

```

public ActionResult Index()
{
    // Define the array of JSON
    List<object> items = new List<object>();
    items.Add(new
    {
        text = "Learn SQL"
    });
    items.Add(new
    {
        text = "Learn PHP"
    });
    items.Add(new
    {
        text = "Learn Bootstrap"
    });
    ViewBag.items = items;
    return View();
}

```

```
}
```

SplitButton

To initialize SplitButton component refer [SplitButton Getting Started documentation](#).

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("Cut").Content("Cut").Render()
    @Html.EJS().Button("Copy").Content("Copy").Render()

    @Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)ViewBag.items).Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    // Define the array of JSON
    List<object> items = new List<object>();
    items.Add(new
    {
        text = "Paste"
    });
    items.Add(new
    {
        text = "Paste Text"
    });
    items.Add(new
    {
        text = "Paste Special"
    });
    ViewBag.items = items;
    return View();
}
```

Note: [View Sample in GitHub](#).

See also

- [Show ButtonGroup selected state on initial render](#)

Accessibility in Button Group Control

The Button group component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Button group component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

The Button group component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Button group component.

Normal behavior

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Focuses the next button in the ButtonGroup.** |

| Enter/Space | Activates the focussed button in the ButtonGroup. |

Checkbox behavior

| Press | To do this |

| --- | --- |

| Tab | Focuses the next button in the ButtonGroup. |

| Space | Activates the focussed button in the ButtonGroup. |

Radiobutton behavior

| Press | To do this |

| --- | --- |

| Tab | Focuses the active button in the ButtonGroup. |

| Right | Activates next/previous button in the ButtonGroup. |

Ensuring accessibility

The Button group component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Button group component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Button group component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Styles and Appearances

To modify the ButtonGroup appearance, you need to override the default CSS of ButtonGroup component. Find the list of CSS classes and its corresponding section in ButtonGroup. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ----- | ----- |

| .e-btn | To customize the ButtonGroup. |

| .e-btn: hover | To customize the ButtonGroup on hover. |

| .e-btn: focus | To customize the ButtonGroup on focus. |

| .e-btn: active | To customize the ButtonGroup on active. |

How To

Create ButtonGroup with icons

ButtonGroup with icons can be achieved by `IconCss` property of the Button component.

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("left").Content("Left").IconCss("e-icons e-left-
    icon").Render()
```

```

    @Html.EJS().Button("middle").Content("Center").IconCss("e-icons e-
middle-icon").Render()
    @Html.EJS().Button("right").Content("Right").IconCss("e-icons e-right-
icon").Render()
</div>
<style>
.e-btn-group {
    margin: 25px 5px 20px 20px;
}
.e-left-icon::before {
    content: '\e33a';
}
.e-right-icon::before {
    content: '\e34d';
}
.e-middle-icon::before {
    content: '\e35e';
}
</style>

```

DEFAULT.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Create ButtonGroup with rounded corner

The ButtonGroup with rounded corner has round edges on both sided. To ButtonGroup with rounded corner, `e-round-corner` class is to be added to the target element.

CSHTML

```

<div class='e-btn-group e-round-corner'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>

```

DEFAULT.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Disable in Button Group Control

Particular button

To disable a particular button in a ButtonGroup, `disabled` attribute should be added to corresponding button element.

Whole ButtonGroup

To disable whole ButtonGroup, `disabled` attribute should be added to all the button elements.

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Disabled(true).Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>
<div class='e-btn-group'>
    @Html.EJS().Button("html_1").Content("HTML").Disabled(true).Render()
    @Html.EJS().Button("css_1").Content("CSS").Disabled(true).Render()

    @Html.EJS().Button("javascript_1").Content("Javascript").Disabled(true).Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: To disable radio/checkbox type ButtonGroup, the `disabled` attribute should be added to the particular input element.

Note: [View Sample in GitHub.](#)

Enable ripple in Button Group Control

Ripple can be enabled by importing `rippleEffect` method from `ej2-base` and initialize `rippleEffect` with `.e-btn-group` element, and selector as `e-btn`.

CSHTML

```
<div class='e-btn-group'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>
<script>
ej.base.enableRipple(true);
var button = document.querySelector('.e-btn-group');
ej.base.rippleEffect(button, {selector: 'e-btn'});
</script>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Enable RTL in Button Group Control

ButtonGroup supports RTL functionality. This can be achieved by adding `e-rtl` class to the target element.

CSHTML

```
<div class='e-btn-group e-rtl'>
    @Html.EJS().Button("html").Content("HTML").Render()
    @Html.EJS().Button("css").Content("CSS").Render()
    @Html.EJS().Button("javascript").Content("Javascript").Render()
</div>
```

DEFAULT.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Form submit in Button Group Control

The name attribute of the input element is used to group the radio or checkbox type ButtonGroup. When the radio or checkbox type are grouped in the form, the checked items value attribute will be posted to the server on form submit that can be retrieved through the name. The disabled radio or checkbox type value will not be sent to the server on form submit.

In the following code snippet, the radio type ButtonGroup is explained with male value as checked. Now, the value that is in checked state will be sent on form submit.

CSHTML

```
<form>
<div class='e-btn-group'>
    <input type="radio" id="male" name="gender" value="male" checked/>
    <label class="e-btn" for="male">Male</label>
    <input type="radio" id="female" name="gender" value="female"/>
    <label class="e-btn" for="female">Female</label>
    <input type="radio" id="transgender" name="gender" value="transgender"/>
    <label class="e-btn" for="transgender">Transgender</label>
</div>
<button class='e-btn e-primary'>Submit</button>
</form>
<style>
.e-btn-group, button {
    margin: 20px 5px 20px 20px;
}
</style>
```

DEFAULT.CS

```
public ActionResult Index()  
{  
    return View();  
}
```

Note: [View Sample in GitHub.](#)

Initialize ButtonGroup using util function

Though, it is a CSS component for easy initialization of ButtonGroup `createButtonGroup` util function can be used.

Using `createButtonGroup` method, the Button options, selector, and cssClass is passed and the corresponding classes is added to the elements.

Create basic ButtonGroup

To create a basic ButtonGroup, the target element along with the button elements should be created.

Create radio type ButtonGroup

To create a radio type ButtonGroup, the target element along with the input elements should be created with type `radio`.

Create checkbox type ButtonGroup

Checkbox type ButtonGroup creation is similar to radio type ButtonGroup, instead the type of the input elements should be `checkbox`.

The following example illustrates how to create ButtonGroup using `createButtonGroup` method for basic, checkbox, and radio type behaviors.

CSHTML

```
<h5>Normal behavior</h5>  
<div id='basic'>  
    <button></button>  
    <button></button>  
    <button></button>  
</div>  
<h5>Checkbox type behavior</h5>  
<div id='checkbox'>  
    <input type="checkbox" id="checkbold" name="font" value="bold"/>  
    <input type="checkbox" id="checkitalic" name="font" value="italic"/>  
    <input type="checkbox" id="checkunderline" name="font"  
value="underline"/>  
</div>  
<h5>Radiobutton type behavior</h5>  
<div id='radio'>  
    <input type="radio" id="radioleft" name="align" value="left"/>  
    <input type="radio" id="radiomiddle" name="align" value="middle"/>  
    <input type="radio" id="radiatoright" name="align" value="right"/>  
</div>  
<script>  
ej.splitbuttons.createButtonGroup('#basic', {  
    buttons: [  

```

```

        { content: 'HTML' },
        { content: 'CSS' },
        { content: 'Javascript' }
    ]
});
ej.splitbuttons.createButtonGroup('#checkbox', {
    buttons: [
        { content: 'Bold' },
        { content: 'Italic' },
        { content: 'Undeline' }
    ]
});
ej.splitbuttons.createButtonGroup('#radio', {
    buttons: [
        { content: 'Left' },
        { content: 'Center' },
        { content: 'Right' }
    ]
});
</script>

```

DEFAULT.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: If null value is passed in button options, then that particular button will be skipped from processing in `createButtonGroup` util function.

Note: [View Sample in GitHub](#).

Show ButtonGroup selected state on initial render

To show selected state on initial render, `checked` property should be added to the corresponding input element.

CSHTML

```

<div class='e-btn-group'>
    <input type="checkbox" id="checkbold" name="font" value="bold" checked/>
    <label class="e-btn" for="checkbold">Bold</label>
    <input type="checkbox" id="checkitalic" name="font" value="italic" />
    <label class="e-btn" for="checkitalic">Italic</label>
    <input type="checkbox" id="checkline" name="font" value="underline"/>
    <label class="e-btn" for="checkline">Underline</label>
</div>

```

DEFAULT.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Button

Getting Started with ASP.NET MVC Button Control

This section briefly explains about how to include [ASP.NET MVC Button](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
```



```
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

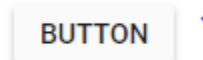
Add ASP.NET MVC Button control

Now, add the Syncfusion ASP.NET MVC Button control in `~/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().Button("element").Content("Button").Render()
```

Press `Ctrl+F5` (Windows) or `⌘+F5` (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Button control will be rendered in the default web browser.



Change Button type

To change the default Button to flat Button, set the `cssClass` property to `e-flat`.

CSHTML

```
@Html.EJS().Button("element").Content("Button").CssClass("e-flat").Render()
```

Note: [View Sample in GitHub.](#)

See also

- [Types of Button](#)

Types and Styles in Button Control

This section explains the different styles and types of Buttons.

Button styles

The Essential JS 2 Button has the following predefined styles that can be defined using the [cssClass](#) property.

Class	Description
-----	-----
e-primary	Used to represent a primary action.
e-success	Used to represent a positive action.
e-info	Used to represent an informative action.
e-warning	Used to represent an action with caution.
e-danger	Used to represent a negative action.
e-link	Changes the appearance of the Button like a hyperlink.

CSHTML

```
@Html.EJS().Button("primarybtn").Content("Primary").CssClass("e-primary").Render()
@Html.EJS().Button("successbtn").Content("Success").CssClass("e-success").Render()
@Html.EJS().Button("infobtn").Content("Info").CssClass("e-info").Render()
@Html.EJS().Button("warningbtn").Content("Warning").CssClass("e-warning").Render()
@Html.EJS().Button("dangerbtn").Content("Danger").CssClass("e-danger").Render()
@Html.EJS().Button("linkbtn").Content("Link").CssClass("e-link").Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
</style>
```

STYLES.CS

```
public ActionResult Styles()
{
    return View();
}
```



Note: Predefined Button styles provide only the visual indication. So, Button content should define the Button style for the users of assistive technologies such as screen readers.

Button types

The types of Essential JS 2 Button are as follows:

- Basic types
- Flat Button
- Outline Button
- Round Button
- Toggle Button

Basic types

The basic Button types are explained below.

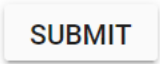
Type	Description
-----	-----
Button	Defines a click Button.
Submit	This Button submits the form data to the server.
Reset	This Button resets all the controls to their initial values.

CSHTML

```
@{IDictionary<string, object> submit = new Dictionary<string, object>();
    submit.Add("type", "submit");
}
@{IDictionary<string, object> reset = new Dictionary<string, object>();
    reset.Add("type", "reset");
}
@Html.EJS().Button("submit").Content("Submit").HtmlAttributes(submit).Render()
@Html.EJS().Button("reset").Content("Reset").HtmlAttributes(reset).Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
</style>
```

BASICBUTTON.CS

```
public ActionResult BasicButton()
{
    return View();
}
```

SUBMITRESET

Flat Button

The Flat Button is styled with no background color. To create a flat Button, set the [cssClass](#) property to `e-flat`.

Outline Button

An outline Button has a border with transparent background. To create an outline Button, set the [cssClass](#) property to `e-outline`.

Round Button

A round Button is shaped like a circle. Usually, it contains an icon representing its action. To create a round Button, set the [cssClass](#) property to `e-round`.

CSHTML

```
@Html.EJS().Button("flat").Content("Flat").CssClass("e-flat").Render()
@Html.EJS().Button("outline").Content("Outline").CssClass("e-
outline").Render()
@Html.EJS().Button("round").IconCss("e-icons e-plus-icon").CssClass("e-
round").IsPrimary(true).Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
.e-plus-icon::before {
    content: '\e823';
}
</style>
```

BUTTONTYPES.CS

```
public ActionResult ButtonTypes()
{
    return View();
}
```

FLAT

OUTLINE



Toggle Button

A toggle Button allows you to change between the two states. The Button is active in toggled state and can be recognized through the `e-active` class. The functionality of the toggle Button is handled by click event. To create a toggle Button, set the [isToggle](#) property to `true`. In the following code snippet, the toggle Button text changes to play/pause based on the state of the Button with the use of click event.

CSHTML

```
@Html.EJS().Button("togglebtn").Content("Play").Click("clickHandler").CssClass("e-flat").IconCss("e-btn-sb-icons e-play-icon").IsToggle(true).Render()
<style>
    @@font-face {
        font-family: 'button-icons';
```

```

src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjluSf8AAAEoAAAAVmNtYXDOXM6wAAABtAAAAFRnbHlmcV/
SKgAAAIQAAAJAaGVhZBNT0QcAAADQAAANmhoZWEIUQQAARAAAAACRobXR4NAAAAAAYAAAA
0bG9jYQNWAA+AAAAIIAAAAHG1heHABGQAZAAABCAAAACBuYW1lASVfhQAABGQAAAJhcG9zdFAouWk
AAABIAAAA2AABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAADQABAAAAQAAYD3WXF8
PPPUACwQAAAAAANGtxgsAAAAA2C3GCwAAAAAD9AP0AAAAACAACAAAAAAAAAAAAEAAAAANAA0AAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wHnDQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAIAAADAAAAFAA
DAAEAAAAUAAQAAAAAYABAABALnCOcN//8AAOcB5wr//wAAAAAAQAGABQAAAABAAMABAHAHAI
ACgAJAAGABQAGAAASADAAAAAADgAkAEQAWgByAIoApGDAAOAA+AEMASAAQAAAAADYQP0AAIAADc
JAZ4Cxp08DAH0AfQAAAIAAAAA9QD9AADAACACUHESEBIREhAm4BZv6a/b4BZv6aDAPo/BgD6AA
AAgAAAAADpwp0AAMADAAANyElISUBBwkBJwERI1kDTvyyAYH+4y4BeQGANv7UTAxNlwEIPf6eAWI
9/ukDEwAAAAIAAAAA/QDngADAACAADchNSETAYEBDAPo/Bj6+gPo/gxipgFy/t0CRwAAAQAAAAA
D9AP0AAsAAAEhFSERMxEhNSERIWHC/koBtnwBtv5KfAI+fp5KAbZ8AbYAAQAAAAAD9AP0AAsAAAE
hFSERMxEhNSERIWHh/isB1T4B1f4rPgIfPv4rAdU+AdUAAgAAAAAD9A0LAAMADAAANyElISUnBxc
3JwCIRIwD6PWyAcWjLO7uLKI/Wj+hoSvs6iyhAm0AAAABAAAAAAP0A/QACwAAAREhFSERMxEhNSE
RAeH+KwHVPgHV/isD9P4rPv4rAdU+AdUAAAAAAGAAAAADdwP0AAMADAAANyElISUBBwkBJwERI4k
C7v0SAVj+0SkBdgF4Kf7RPgw+rQEJL/64AUgv/vgC/AAAAEAAAAA/QD9AALAAABIRUHEtMRITU
hESMB2v4yAc5Mac7+MkwCJkz+MgHOTAH0AAIAAAAA/QDzQADAACAADchNSE1KQEBDAPo/BgB9AH
0/gwzpzUCYAACAAAAAAP0A80AAwAHAAA3ITUhNSkBAQwD6PwYafQB9P4MM6WVAmAAAAASAN4AAQA
AAAAAABAAAAAQAAAAAQAMAAEAQAAAAAAGAHAA0AAQAAAAAAAwAMABQAAQAAAAABAAMACA
AAQAAAAABQALACwAAQAAAAABGAMADCAQAAAAACGAsAEMAAQAAAAACWASAG8AAwABBakAAAA
CAIEAAwABBakAAQAYAIMAAwABBakAAgAOAJsAAwABBakAAwAYAKkAAwABBakABAAYAMEAAwABBak
ABQAWANKAAwABBakABgAYA08AAwABBakACgBYAQCAAwABBakACwAkAV8gYnV0dG9uLW1jb25zUmV
ndWxhcmJldHRvbilpY29uc2JldHRvbilpY29uc1ZlcnNpb24gMS4wYnV0dG9uLW1jb25zRm9udCB
nZW5lcmF0ZWQgdXNpbmcgU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20
AIABIAHUADAB0AG8AbgAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGIAdQB0AHQAbwBuAC0AaQB
jAG8AbgBzAGIAdQB0AHQAbwBuAC0AaQBjAG8AbgBzAFYAZQByAHMAaQBvAG4AIAAxAC4AMABIAHU
AdAB0AG8AbgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQAIAB1AHMAaQB
uAGCAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHc
ALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAKAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAANAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4ACm1lZG1hLXBsYXkLbWVkaWEtcGF
1c2UQLWRvd25sb2FkLTAyLXdmLQltZWRpYS1lbmQHYWRkLW5ldwtuZXctbWFpbC13Zhb1c2VyLWR
vd25sb2FkLXdmDGv4cGFuZC0wMy13Zg5kb3dubG9hZC0wMi13ZgphZGQtbmV3XzAxX2C21lZG1hLWV
qZWNOdm1lZG1hLWVqZWNOLTAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-btn-sb-icons {
font-family: 'button-icons';
line-height: 1;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-play-icon::before {
content: '\e701';
}
.e-pause-icon::before {
content: '\e705';
}
}
</style>
<script>

```

```
function clickHandler() {
    var togglebtn =
ej.base.getComponent(document.getElementById("togglebtn"), 'btn');
    if
(document.getElementById('togglebtn').classList.contains('e-active')) {
        togglebtn.content = 'Pause';
        togglebtn.iconCss = 'e-btn-sb-icons e-pause-icon';
    } else {
        togglebtn.content = 'Play';
        togglebtn.iconCss = 'e-btn-sb-icons e-play-icon';
    }
}
</script>
```

TOGGLEBUTTON.CS

```
public ActionResult ToggleButton()
{
    return View();
}
```

▶ PLAY

Icons

Button with font icons

The Button can have an icon to provide the visual representation of the action. To place the icon on a Button, set the [iconCss](#) property to `e-icons` with the required icon CSS. By default, the icon is positioned to the left side of the Button. You can customize the icon's position by using the [iconPosition](#) property.

CSHTML

```
@Html.EJS().Button("iconbutton").Content("PREVIOUS").IconCss("e-btn-sb-icon
e-prev-icon").Render()
@Html.EJS().Button("iconposbtn").Content("STOP").IconCss("e-btn-sb-icon e-
stop-icon").Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
@@font-face {
    font-family: 'btn-icon';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAwAgT1MvMj1tSfgAAAEoAAAAVmNtYXDnH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAAQYaGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAYAAAA
gbG9jYQON6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAACAAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaawAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAEABAAAAEAAgADAAQABQAGAACAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAlIREhASERIQJpAV7+ov3
```

```

QAV7+ogwD6PwYA+gAAAAEAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00Df0mDQ4NDawLCwkJCAcFBQMCA239Jg4NDQ0LCwsJCQgHBQUdAgEBAgMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JBjU/BDUvCyMPAQytrQH5AgoEAQEBAg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAgIEBAoBAQEECGcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwKEAwIBAQMDcGwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCaKICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAAzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXMz8KNScxBxEfDjSBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DYtIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQL7Y0EAwQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBWGCACaHBWYGBgUFBAQDAGIBAQEBAgI
DBAQFBQYGBgcHBwgMAQMDawUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDawTV+94BAQECawMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBWYGBQQEAWI
CUAgHBwcGBgYFBQQEAWMBAgIBAwMEBAUFBQcGBwCHCAImCAcHBWYGBgUFBAQDAGIBAdUJCQgICAg
GBWYFBAQDAGEBAAAAAIAAAAAA6cD9AADAaWaadchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EWMTZcBCD3+ngFiPf7pAXMAAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAgAAQABAAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAgAKWABAAA
AAAAKACwAMwABAAAAAALABIAxWADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMabwBuAFIAZQBnAHUAbABhAHIAyB0AG4ALQBpAGM
AbwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMabwB
uAEYAbwBuAHQAIABnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBAGMAZgB1AHM
AaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAAGBAGEDAQQBBQE
GAQcBCAEJAaptZWRpYS1wbGF5C2l1ZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAXBGNvcHkQLWRvd25sb2FkLTAyLXdmLQAA) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
/*For Right Icon Button*/
.e-stop-icon::before {
    content: '\e790';
}
/*For Left Icon Button*/
.e-prev-icon::before {
    content: '\e716';
}
}
</style>

```

BUTTONICON.CS

```
public ActionResult ButtonIcon()  
{  
    return View();  
}
```

[< PREVIOUS](#)[STOP >](#)

Note: The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icons on the Button using the [iconCss](#) property.

Button with SVG image

SVG image can be added to the Button using [iconCss](#) property.

In the following example, SVG image is added using the iconCss class `e-search-icon` by setting `height` and `width` property.

CSHTML

```
@Html.EJS().Button("iconbutton").IconCss("e-search-icon").Render()  
<style>  
button {  
    margin: 25px 5px 20px 20px;  
}  
.e-btn-icon.e-search-icon {  
    background: url('search_icon.svg');  
    height: 25px;  
    width: 25px;  
}  
</style>
```

BUTTONSVG.CS

```
public ActionResult ButtonSvg()  
{  
    return View();  
}
```



Button size

The two types of Button sizes are default and small. To change the size of the default Button to small Button, set the [cssClass](#) property to `e-small`.

CSHTML

```
@Html.EJS().Button("smallbtn").Content("SMALL").CssClass("e-small").Render()
```



```
@Html.EJS().Button("normalbtn").Content("NORMAL").Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
</style>
```

SIZE.CS

```
public ActionResult Size()
{
    return View();
}
```



See Also

- [Customize Button appearance](#)
- [How to create block button](#)
- [How to create repeat button](#)

Styles and Appearances

To modify the Button appearance, you need to override the default CSS of Button component. Find the list of CSS classes and its corresponding section in Button component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

- |.e-btn|To customize the button.
- |.e-btn: hover|To customize the button on hover.
- |.e-btn: focus|To customize the button on focus.
- |.e-btn: active|To customize the button on active.

Accessibility in Button Control

The Button component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

```
| Section 508 Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| Accessibility Checker Validation |  |

| Axe-core Accessibility Validation |  |

<style>

.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}

</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Button component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Button component:

```
| Attributes | Purpose |
| --- | --- |
| aria-label | Provides an accessible name for the icon only button. |
```

Keyboard interaction

The Button component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Button component.

| **Press** | **To do this** |

| --- | --- |

| **Space** | When the button has focus, pressing the space key changes the state of the button. |

Ensuring accessibility

The Button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Button component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

How To

Add link to a Button

The appearance of the Button can be changed like a hyperlink by `e-link` class using [cssClass](#) property and link navigation can be handled in Button click event.

In the following example, link is added in Button click event by using `window.open()` method.

CSHTML

```
@Html.EJS().Button("button").CssClass("e-link").Content("Go to  
Google").Render()  
<style>  
button {  
    margin: 25px 5px 20px 20px;  
}  
</style>  
<script>  
document.getElementById('button').onclick = (): void => {  
    window.open("https://www.google.com");  
};  
</script>
```

LINK.CS

```
public ActionResult link()  
{  
    return View();  
}
```

Note: [View Sample in GitHub.](#)

Create a Block Button

You can customize a Button into a Block Button that will span the entire width of its parent element. To create a Block Button, set the [cssClass](#) property to `e-block`.

CSHTML

```
@Html.EJS().Button("blockbtn").Content("BLOCKBUTTON").CssClass("e-block").Render()
@Html.EJS().Button("primarybtn").Content("BLOCKBUTTON").CssClass("e-block").IsPrimary(true).Render()
@Html.EJS().Button("successbtn").Content("BLOCKBUTTON").CssClass("e-block e-success").Render()
<style>
button {
    margin: 25px 0;
}
</style>
```

BLOCKBUTTON.CS

```
public ActionResult blockButton()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Customize Button Appearance

You can customize the appearance of the Button by using the Cascading Style Sheets (CSS). Define the CSS according to your requirement, and assign the class name to the [cssClass](#) property. In the following code snippet the background color, text color, height, width, and sharp corner of the Button can be customized through the `e-custom` class for all states (hover, focus, and active).

CSHTML

```
@Html.EJS().Button("custom").Content("CUSTOM").CssClass("e-custom").Render()
<style>
.e-custom {
    border-radius: 0;
    height: 30px;
    width: 80px;
}
.e-custom, .e-custom:hover, .e-custom:focus, .e-custom:active {
    background-color: #ff6e40;
    color: #fff;
}
button {
    margin: 25px 5px 20px 20px;
}
</style>
```

CUSTOMBUTTON.CS

```
public ActionResult CustomButton()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Customize input and anchor elements

You can customize the appearance of the input and anchor elements using predefined styles through the class property. In the following code snippet, the input element is customized as a link Button by setting the e-btn e-link class, and the anchor element is customized as a primary Button by setting the e-btn e-primary class.

CSHTML

```
<div>
    <input type="button" id="inputbtn" value="Input Button" class="e-btn e-
link">
</div><br>
<div>
    <a id="anchorbtn" class="e-btn e-primary" href="#">Google</a>
</div>
```

CUSTOM-ELEMENT.CS

Note: [View Sample in GitHub.](#)

Repeat Button in Button Control

The Repeat button is a type of Button in which the click event is triggered at regular time interval from the pressed state till the released state.

The following example explains about how to achieve Repeat Button in mouse and touch events.

CSHTML

```
<div class='btncontainer'>

@Html.EJS().Button("button").Content("Button").Click("clickHandler").Render(
)
</div>
<div class='event' style="height:auto;">
    <table title='Event Trace' style="width:100%">
        <tbody>
            <tr>
                <th>Event Trace</th>
            </tr>
            <tr>
                <td>
                    <div class="eventarea" style="height: 250px;overflow: auto">
                        <span id="eventlog" style="word-break: normal;"></span>
                    </div>
                </td>
            </tr>
            <tr>
                <td>
                    <div class="evtbtn" style="padding:20px 0 0 20px">

@Html.EJS().Button("clear").Content("Clear").Click("clearHandler").Render()
                    </div>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

```

    </tr>
</tbody>
</table>
</div>
<style>
hr {
    margin: 1px 10px 1px 0px;
    border-top: 1px solid #eee;
}
.btncontainer {
    float: left;
    width: 40%;
}
.event {
    float: right;
    width: 60%;
    border-left: 1px solid #D7D7D7;
}
#eventlog b {
    color: #388e3c;
}
</style>
<script>
    document.addEventListener("DOMContentLoaded", function () {
        document.getElementById("button").addEventListener('mousedown',
function () {
    event.preventDefault();
    timeout = setInterval(clickEventHandler, 200);
});
        document.getElementById("button").addEventListener('touchstart',
function () {
    event.preventDefault();
    timeout = setInterval(clickEventHandler, 200);
});
        document.getElementById("button").addEventListener('mouseup',
function () {
    clearInterval(timeout);
});
        document.getElementById("button").addEventListener('touchend',
function () {
    clearInterval(timeout);
});
    });

    function clickHandler() {
        appendSpanElement('Button click event triggered.<hr>');
    }
    function clearHandler() {
        document.getElementById('eventlog').innerHTML = '';
    }
    function clickEventHandler(e) {
        appendSpanElement('Button click event triggered.<hr>');
    }
    function appendSpanElement(text) {
        var span = document.createElement('span');
        span.innerHTML = text;
        var log = document.getElementById('eventlog');

```

```
log.insertBefore(span, log.firstChild);
}
</script>
```

REPEATBUTTON.CS

```
public ActionResult RepeatButton()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Right to Left in Button Control

Button component has RTL support. This can be achieved by setting [enableRtl](#) as true.

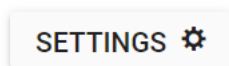
The following example illustrates how to enable right-to-left support in Button component.

CSHTML

```
@Html.EJS().Button("rtl").Content("Settings").IconCss("e-icons e-setting-
icon").EnableRtl(true).Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
.e-setting-icon::before {
    content: '\e7cf';
}
</style>
```

RTL.CS

```
public ActionResult rtl()
{
    return View();
}
```



Note: [View Sample in GitHub.](#)

Set the disabled state

Button component can be enabled or disabled by giving [disabled](#) property. To disable Button component, the `disabled` property can be set as true.

The following example demonstrates Button in disabled state.

CSHTML

```
@Html.EJS().Button("disabled").Content("Disabled").Disabled(true).Render()
```

```
<style>
button {
    margin: 25px 5px 20px 20px;
}
</style>
```

DISABLEDBUTTON.CS

```
public ActionResult DisabledButton()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Tooltip for Button Control

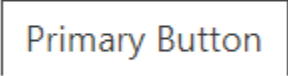
Tooltip can be shown on Button hover and it can be achieved by setting `title` attribute.

CSHTML

```
@Html.EJS().Button("primarybtn").HtmlAttributes(ViewBag.Attributes).Content(
"Button").IsPrimary(true).Render()
<style>
button {
    margin: 25px 5px 20px 20px;
}
</style>
```

TOOLTIP.CS

```
public ActionResult tooltip()
{
    Dictionary<string, object> Attributes = new Dictionary<string, object>()
    {
        { "title", "Primary Button" }
    };
    ViewBag.Attributes = Attributes;
    return View();
}
```



Note: [View Sample in GitHub.](#)

Migration from Essential JS 1

This article describes the API migration process of Button component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Specifies the text of the button | **Property:** *text*

@Html.EJ().Button("btn").Text("Button") | **Property:** *content*

@Html.EJS().Button("btn").Content("Button").Render() |

| Specifies the content type of the button | **Property:** *ContentType*

@Html.EJ().Button("btn").Text("Save").ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-save") | Not applicable |

| Specifies icon for the button | **Property:** *prefixIcon*

@Html.EJ().Button("btn").ContentType(ContentType.ImageOnly).PrefixIcon("e-icon e-save") |

Property: *iconCss*

 @Html.EJS().Button("btn").IconCss("e-icons e-save").Render() |

| Positioning icon in the button | **Property:** *imagePosition*

@Html.EJ().Button("btn").Text("Save").ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-save").ImagePosition(ImagePosition.ImageRight) | **Property:** *iconPosition*

@Html.EJS().Button("btn").Content("Save").IconCss("e-icons e-save").IconPosition(Syncfusion.EJ2.Buttons.IconPosition.Right).Render() |

| Specifies secondary icon for the button | **Property:** *suffixIcon*

@Html.EJ().Button("btn").Text("Save").ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-save").SuffixIcon("e-icon e-file-html") | Not applicable |

| Adding custom class | **Property:** *cssClass*

@Html.EJ().Button("btn").Text("Button").Size(ButtonSize.Small).CssClass("custom-class") |

Property: *cssClass*

 @Html.EJS().Button("btn").Content("Button").CssClass("custom-class").Render() |

| Specifies the size of the button | **Property:** *size*

@Html.EJ().Button("btn").Text("Button").Size(ButtonSize.Small) | **Property:** *cssClass*

@Html.EJS().Button("btn").Content("Button").CssClass("e-small").Render() |

| Triggers click event repeatedly while pressing the button | **Property:** *repeatButton*

@Html.EJ().Button("btn").Text("Click").RepeatButton(true) | Not applicable |

| Sets time interval between two consecutive click event on the repeat button | **Property:** *timeInterval*

 @Html.EJ().Button("btn").Text("Click").RepeatButton(true).TimeInterval("100") | Not applicable |

| Specifies the type of the button | **Property:** *type*

@Html.EJ().Button("btn").Text("Submit").Type(ButtonType.Submit) | Not applicable |

| Changes normal button into rounded corner | **Property:** *showRoundedCorner*

@Html.EJ().Button("btn").Text("Button").ShowRoundedCorner(true) | Not applicable |

| Specifies the width of the button | **Property:** *width*

@Html.EJ().Button("btn").Text("Button").Width("150px") | Not applicable |

| Specifies the height of the button | **Property:** *height*

@Html.EJ().Button("btn").Text("Button").Height("50px") | Not applicable |

| Adds HTML attributes to the button | **Property:** *htmlAttributes*

@Html.EJ().Button("btn").Text("Button").HtmlAttributes("") | Not applicable |

| Allows the appearance of the Button to be enhanced and visually appealing | Not applicable |
Property: *isPrimary*

@Html.EJS().Button("btn").Content("Button").IsPrimary(true).Render() |

| Makes the button toggle from normal to active state | **Property:** *isToggle*

 Not applicable |
Property: *isToggle*

@Html.EJS().Button("btn").Content("Button").IsToggle(true).Render() |

| Specifies the disabled state of the button | **Property:** *enabled*

@Html.EJ().Button("btn").Text("Button").Enabled(false) | **Property:** *disabled*

@Html.EJS().Button("btn").Content("Button").Disabled(true).Render() |

| Enable or disable rendering component in right to left direction | **Property:** *enableRTL*

@Html.EJ().Button("btn").Text("Save").ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-save").EnableRtl(true) | **Property:** *enableRtl*

@Html.EJS().Button("btn").Content("Save").IconCss("e-icon e-save").EnableRtl(true).Render() |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroys the control | **Methods:** *destroy*

 @Html.EJ().Button("btn").Text("Button")

 var btnObj = \$("#btn").data("ejButton");
btnObj.destroy(); | **Methods:** *destroy*

@Html.EJS().Button("btn").Content("Button").Render()
 var btnObj =
document.getElementById("btn").ej2_instances[0];
btnObj.destroy(); |

| Disables the button | **Methods:** *disable*

 @Html.EJ().Button("btn").Text("Button")

var btnObj = \$("#btn").data("ejButton");
btnObj.disable(); | Not applicable |

| Enables the button | **Methods:** *enable*

 @Html.EJ().Button("btn").Text("Button")

var btnObj = \$("#btn").data("ejButton");
btnObj.enable(); | Not applicable |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers while clicking the button | **Events:** *click*

@Html.EJ().Button("btn").Text("Button").Click("btnClick")
function btnClick(args) {
/
code block */
} | Not applicable |

| Triggers once the component rendering is completed | **Events:** *create*

@Html.EJ().Button("btn").Text("Button").ClientSideEvents(e => e.Create("create"))
function
create(args) {
/ code block /
} | **Events:** *created*


```
@Html.EJS().Button("btn").Content("Button").Created("created").Render()<br/>function created()
{<br/>/ code block </br/>} |
```

| Triggers once the component is destroyed | **Events:** *destroy*


```
@Html.EJ().Button("btn").Text("Button").Destroy("destroy") <br/>function destroy(args) {<br/>/
code block */<br/>} | Not applicable |
```

Calendar

Getting Started with ASP.NET MVC Calendar Control

This section briefly explains about how to include [ASP.NET MVC Calendar](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

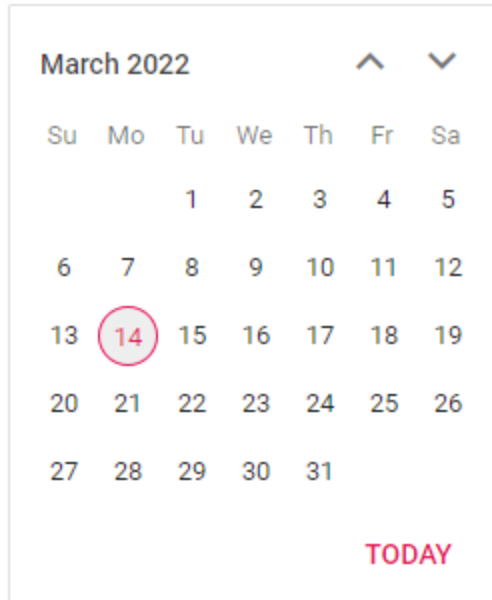
Add ASP.NET MVC Calendar control

Now, add the Syncfusion ASP.NET MVC Calendar control in `~/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().Calendar("calendar").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Calendar control will be rendered in the default web browser.



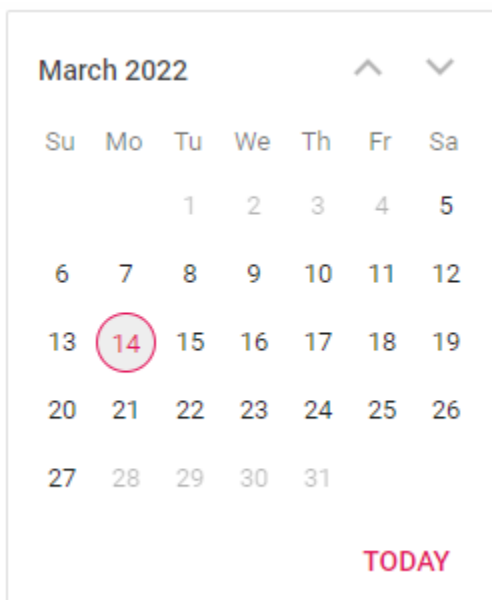
Setting the value within min and max dates

After rendering a simple Calendar control by following the above steps, configure the Calendar to set a value within a specific range using its value, min, and max properties.

Here the Calendar allows you to select a date within the range from 5th to 27th of this month.

CSHTML

```
@{
    .....
    var minDate = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 05);
    var maxDate = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 27);
}
@Html.EJS().Calendar("calendar").Min(minDate).Max(maxDate).Render()
```



Note: [View Sample in GitHub.](#)

See also

- [Select multiple dates in the Calendar](#)
- [Render Calendar with specific culture](#)
- [How to change the initial view of the Calendar](#)
- [Render Calendar with week numbers](#)
- [Show other month dates](#)

Date Range in Calendar Control

Calendar provides an option to select a date value within a specified range by defining the [min](#) and [max](#) properties. The min date should always be lesser than the max date. If the value of [min](#) or [max](#) properties are changed through code behind, then update the [value](#) property to be set within the specified range. Or else, if the value is out of specified date range and less than [min](#) date, value property will be updated with min date or the value is higher than max date, value property will be updated with [max](#) date.

CSHTML

```
@Html.EJS().Calendar("element").Value(ViewBag.value).Min(ViewBag.minDate).Max(ViewBag.maxDate).Render()
```

DATERANGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult DateRange()
        {
            ViewBag.minDate= new
            DateTime(DateTime.Now.Year,DateTime.Now.Month,07);
            ViewBag.maxDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 27);
            ViewBag.value = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 14);
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Multi Selection in Calendar Control

Calendar provides an option to select **single** or **multiple dates** by using [isMultiSelection](#) and [values](#) properties. By default, [isMultiSelection](#) property will be in disabled state.

API	Type	Description
isMultiSelection	Boolean	Enables the multi-selection option in the Calendar control
values	Date[]	Gets or sets the date range values in multi-selection option

The following example demonstrates the functionality of `isMultiSelection` property and `values` properties in the Calendar control.

CSHTML

```
@Html.EJS().Calendar("calendar").IsMultiSelection(true).Render()
```

MULTI.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult Multi()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Globalization in ASP.NET MVC Calendar Control

Globalization is the combination of adapting the control to various languages by means of parsing and formatting the date or number [Internationalization](#) and also by adding cultural specific customizations and translating the text [localization](#).

By default, Calendar date format, week and month names are specific to American English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by uses the official [UNICODE CLDR](#) JSON data.

- Set the culture by using the [locale](#) property.

To go with the different culture other than `English`, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the [CLDR-Data](#) link.

,

```
npm install cldr-data --save
```

Once the package installed, you can find the culture specific JSON data under the location `\node_modules\cldr-data`.

In ASP.NET MVC refer the culture files directly from `/node_modules/cldr-data` location as like the below code examples.

```
`typescript
function loadCultureFiles(de) {
var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
var loader = ej.base.loadCldr;
var loadCulture = function (prop) {
var val, ajax;

ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-data/main/' +
name + '/' + files[prop], 'GET', false);
ajax.onSuccess = function (value) {
val = value;
};
ajax.send();
loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
loadCulture(prop);
}
}
```

Note: The `Localization` library allows you to localize default text content of the Calendar. The Calendar control has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the `locale` value and translation object.

```
| Locale keywords |Text |
| ----- | ----- |
| today | Name of the button to choose Today date. |
```

- Before changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through `load` method of `L10n` class.

```
`typescript
var L10n = ej.base.L10n;
```



```

L10n.load({
"de": {
"calendar": {
"today": "heute"
}
}
});

```

The following example demonstrates the Calendar in **German** culture.

CSHTML

```

@Html.EJS().Calendar("calendar").Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        calendarObject =
document.getElementById('calendar').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            "de": {
                "calendar": {
                    "today": "heute"
                }
            }
        });
        loadCultureFiles('de');
        calendarObject.locale = 'de';
    });
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
            } else {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
    }

```

```

    };
    for (var prop = 0; prop < files.length; prop++) {
        loadCulture(prop);
    }
}
</script>

```

INTERNATIONAL.CS

Right-To-Left

The Calendar supports right-to-left functionality for languages like Arabic, Hebrew, etc. To display the text in the right-to-left direction, use [enableRtl](#) property.

The following example demonstrates the Calendar in **Arabic** culture with Right-To-Left direction.

CSHTML

```

@Html.EJS().Calendar("calendar").EnableRtl(true).Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        calendarObject =
document.getElementById('calendar').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            "en": {
                "calendar": {
                    "today": "Today"
                }
            }
        });
        loadCultureFiles('ar');
        calendarObject.locale = 'ar';
    });
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
            } else {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            }
        }
    }
}

```

```

    };
    ajax.send();
    loader(JSON.parse(val));
  };
  for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}
</script>

```

RTL.CS

Customization in Calendar Control

Each day cell of the Calendar can be customized by using the [renderDayCell](#) event.

The following section demonstrates how to disable or highlight specific dates in a Calendar.

Disable weekends

You can disable weekends of every month in a Calendar by using the [renderDayCell](#) event. The `renderDayCell` event offers the following arguments on each day cell creation to help you disable the dates.

View	Description
---	---
<code>date</code>	Defines the current date of the Calendar.
<code>isDisabled</code>	Specifies whether the current date is to be disabled or not.
<code>isOutOfRange</code>	Defines whether the current date is out of range or not.

CSHTML

```

@Html.EJS().Calendar("element").RenderDayCell("disableDate").Value(@ViewBag.
value).Render()
<script>
    function disableDate(args) {
        if (args.date.getDay() === 0 || args.date.getDay() === 6) {
            args.isDisabled = true;
        }
    }
</script>

```

CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{

```

```

public class CalendarController : Controller
{
    public ActionResult sample()
    {
        ViewBag.value = new DateTime(DateTime.Now.Year,
        DateTime.Now.Month, 15);
        return View();
    }
}

```

Day cell format

You can also highlight specific dates by adding custom CSS or element to the day cell by using the [renderDayCell](#) event.

You can customize the appearance of the Calendar by overriding the existing styles. The following list of CSS class names are used to customize the Calendar control.

Class Name	Description
--- ---	
e-calendar	Applied to the Calendar.
e-header	Applied to the header.
e-title	Applied to the title.
e-icon-container	Applied to the previous and next icon container.
e-prev	Applied to the previous icon.
e-next	Applied to the next icon.
e-weekend	Applied to weekends.
e-other-month	Applied to days of other months.
e-day	Applied to each day cell.
e-selected	Applied to the selected dates.
e-disabled	Applied to the disabled dates.

The following example highlights the World Health Day (every 7th April) and World Forest Day (every 21st March) by using the custom icon and ToolTip.

CSHTML

```

@Html.EJS().Calendar("element").Value(@ViewBag.value).RenderDayCell("customD
ates").Render()
<script>
    var addClass = ej.base.addClass;
    function customDates(args) {
        /*Date need to be customized*/
        var span;
        if (args.date.getDate() === 10) {
            span = document.createElement('span');
            span.setAttribute('class', 'e-icons highlight');

```

```

        args.element.firstChild.setAttribute('title', 'Birthday
!');

        addClass([args.element], ['e-day', 'special', 'birthday']);
        args.element.setAttribute('data-val', ' Birthday !');
        args.element.setAttribute('title', 'Birthday !');
        args.element.appendChild(span);
    }
    if (args.date.getDate() === 15) {
        span = document.createElement('span');
        span.setAttribute('class', 'e-icons highlight');
        args.element.firstChild.setAttribute('title', 'Farewell
!');

        addClass([args.element], ['e-day', 'special', 'farewell']);
        args.element.setAttribute('data-val', 'Farewell !');
        args.element.setAttribute('title', 'Farewell !');
        args.element.appendChild(span);
    }
}
</script>
<style>
    span.e-icons.highlight {
        margin-top: -13px;
        display: block;
        margin-left: 4px;
    }

    span.e-icons.highlight,
    span.e-icons.highlight:before {
        color: rgb(0, 0, 255);
    }

    .e-other-month span.e-icons.highlight:before {
        content: "";
    }

    span.e-icons.highlight:before {
        content: "\e190";
        vertical-align: middle;
        margin-right: 3px;
        font-size: 4px;
        position: relative;
        top: -1px;
        font-weight: normal;
    }

    .e-selected span.e-icons.highlight:before {
        color: #fff;
    }

    span.e-icons.highlight,
    .e-customStyle.e-calendar .e-content span.special,
    .e-customStyle.e-calendar .e-content span.daycell {
        font-weight: bold;
    }

    @@font-face {
        font-family: 'e-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQYkCgIAAAIkAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAAQAAIbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAAMAQIAAwAAAAA

```

```

AAgAAAAoACgAAAP8AAAAAQAAPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAGAAAAAUAAMAAQA
AABQABABeAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAGABWAKAAKABQALAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAA+oD6gALAAATCQEXCQEHCEQCEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHESEVIREhFSE
VA9b8KgPW/CoDlvwq6I0B64wB640AAAAEAAAAA+oD4QCaAAABMx8ahQEPDjEPah8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBWYGBgUEBAMCAgECAwUFBggICQoLCwWMDg0GAgEBAGIDBAMIBiI
dHh0cHB0zFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAS
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBWgICAKJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBWgICAKKCgs
LDaXNDQ4ODxAQEBEWFxYWFhYVFRQUEXIRERAOFxMLCggIBgYFBgQMDaWNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEHERERAPDw4NDQsMBWgFBgYICQkKDAwODw8RERMTEUUFhUWFxYWFxEQEBAPDg4
NDQWMCwsKCgkICAgHBgYFBQQEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM
DAWIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAGMDAwQE0AUEAWMDAgEBAGMDAwQ
FfaUEAWMDAgEBAGMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRochR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECawMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAWMDAgEBAGMDAwQFaAQEAwQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxFYVEXEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHB0zFxFYVEXE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAE0AQcuASc+ATceAQEHBgcnJgYPAQYWHWE
GFBChDgEfAR4BPWEWHwEeATsBMjY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8Bji8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhSBhIIrgg
FAziFe4QJDAWJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQWMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VhWMAc0AAAAAAQAAAAADAQPqAAUAAAEEXCQEAQL1Hf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBJv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUFHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFbQTExITEREREbAPDw0ODAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFbQTExITEREREbAPDw0ODAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwWNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYVFRQUEXIREQ8PDg0MCwoJCACFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVfHYWGBcYGRkZGhoZGRkYFhgWfHYVFBQTEHERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBWgJCQsLDAwODQ8PEBARERETEhMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBApDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBWgJCQsLDAwODQ8PEBARERETEhM
TFBQUFRoZGRkYFhgWfHYVFBQTEHERDw8ODQwLCgkIBwUFAwICAwUFBWgJCgsMDQ4PDxEREhMUFBU
WFhYFhgZGRkaGhkZGRGxGBYWFhUUFBMSEREPdW4NDASKCQgHBQUDAgIDBQUHCAkKCwWNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMHw8RDw8hLw8RPw6aAswNDgWMDAsKCgkIBwU
FAWIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICACFBQMCAQECAwUFBWgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDASLCgoICACFBQMCAQECAwUFBWgICgoLCwWNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAA0AAQABAAAAAACAACADgABAAAAA
DAA0AFQABAAAAAAEAA0AIgABAAAAAFAASALwABAAAAAAGAA0AogABAAAAAAKACwARwABAAA
AAAAALABIAcWADAAEECQAAAAIAHQADAAEECQABAB0AhWADAAEECQACAA4AoQADAAEECQADAB0ArWA
DAAEECQAEAB0AyQADAAEECQAFABYA4wADAAEECQAGAB0A+QADAAEECQAKAFgBEWADAAEECQALACQ
BayBlLWlj25zLW1ldHJvUmVndWxhcmtUtaWNvbnMtbnV0cm91LWlj25zLW1ldHJvVmVyc2lvbiA
xLjBlLWlj25zLW1ldHJvRm9udCBnZW51cmF0ZWZgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIBAg0AaAQBJAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4ACwAtAG0AZQB0AHIAbWBlAC0AaQBJAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAAC4AMABlAC0AaQBJAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIAABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAAB
NAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMhbnVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWfYDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('true');

```

```

        font-weight: normal;
        font-style: normal;
    }
</style>

```

CELLFORMAT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult SpecialDates()
        {
            ViewBag.value = new DateTime(2017, 3, 10);
            return View();
        }
    }
}

```

Highlight Weekends

You can highlight the weekends of every month in a Calendar by using the [renderDayCell](#) event.

CSHTML

```

@Html.EJS().Calendar("element").RenderDayCell("highlightDate").Value(@ViewBag.value).Render()
<script>
    function highlightDate(args) {
        if (args.date.getDay() === 0 || args.date.getDay() === 6) {
            // To highlight the week end of every month
            args.element.classList.add('e-highlightweekend');
        }
    }
</script>
<style>
    .e-highlightweekend {
        background-color: #cfe9f3;
    }
</style>

```

HIGHLIGHT-WEEKEND.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {

```

```

{
    public ActionResult sample()
    {
        ViewBag.value = new DateTime(DateTime.Now.Year,
        DateTime.Now.Month, 15);
        return View();
    }
}

```

Note: [View Sample in GitHub.](#)

See also

- [Add the external button in the Calendar popup](#)
- [How to skip a month in Calendar](#)
- [How to change the first day of week](#)
- [How to customize the Calendar day header](#)

Calendar Views in Calendar Control

The Calendar has the following pre-defined views that provide a flexible way to navigate back and forth when selecting dates.

| **View** | **Description** |

| --- | --- |

| month (default) | Displays the days in a month. |

| year | Displays the months in a year. |

| decade | Displays the years in a decade. |

When view is defined to the [start](#) property of the Calendar, it allows you to set the initial view on rendering.

The following example demonstrates how to set the **year** as the start view of the Calendar.

CSHTML

```

@Html.EJS().Calendar("element").Value(@ViewBag.value).Start(Syncfusion.EJ2.C
alendars.CalendarView.Year).Render()

```

VIEWS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult sample()
        {

```



```

        ViewBag.value = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15);
        return View();
    }
}

```

View restriction

By defining the start and depth property with the different view, drill-down and drill-up views navigation can be limited to the user. Calendar views will be drill-down up to the view which is set in [depth](#) property and drill-up to the view which is set in [start](#) property.

The following example displays the Calendar in **decade** view, and allows you to select a date in **month** view.

Note: Depth view should always be smaller than the start view. If the **depth** and **start** views are the same, then the Calendar view remains unchanged.

CSHTML

```

@Html.EJS().Calendar("element").Value(@ViewBag.value).Depth(Syncfusion.EJ2.Calendars.CalendarView.Year).Start(Syncfusion.EJ2.Calendars.CalendarView.Decade).Render()

```

RESTRICTION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult sample()
        {
            ViewBag.value = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 15);
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

Accessibility in Calendar Control

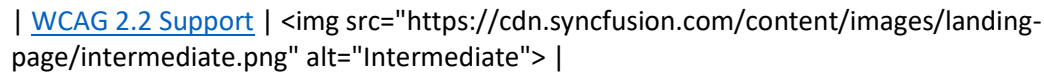
The web accessibility makes web content and web applications more accessible for disabled people. It especially helps in dynamic content change and development of advanced user interface controls with AJAX, HTML, JavaScript, and related technologies.

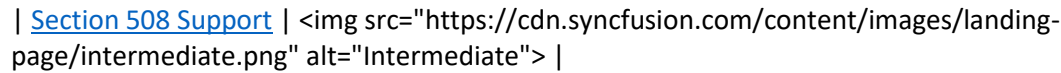
The Calendar component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

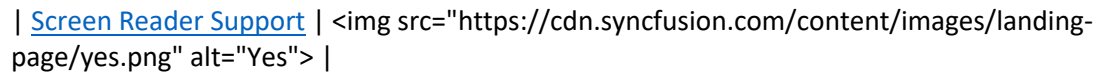
The accessibility compliance for the Calendar component is outlined below.

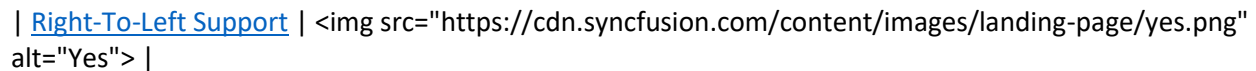
| [Accessibility Criteria](#) | [Compatibility](#) |

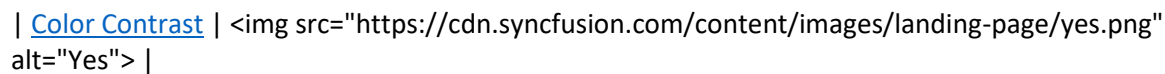
| -- | -- |

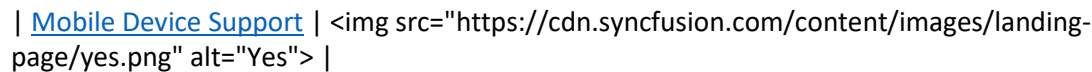
| [WCAG 2.2 Support](#) |  |

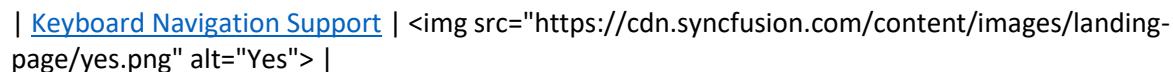
| [Section 508 Support](#) |  |

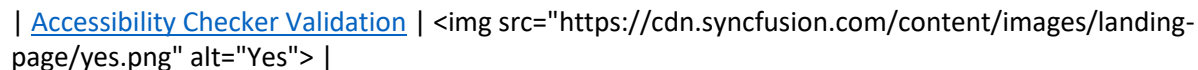
| [Screen Reader Support](#) |  |

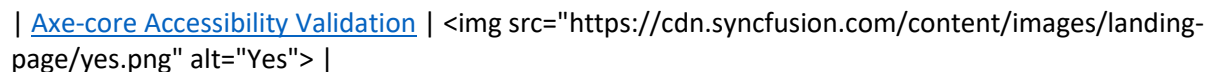
| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

Calendar provides built-in compliance with [WAI-ARIA](#) specifications. WAI-ARIA support is achieved through attributes like `aria-label`, `aria-selected`, `aria-disabled`, and `aria-activedescendant` applied for navigation buttons, and disable and active day cells.

It helps disabled persons by providing information about the widget for assistive technology in the screen readers. Calendar control contains grid role and grid cell for each day cell.

- **Aria-label:** This attribute provides text labels for an object for the previous and next month's elements. It helps the screen reader object to read.
- **Aria-selected:** Indicates the currently selected date of the Calendar control.
- **Aria-disabled:** Indicates the disabled state of the Calendar control.
- **Aria-activedescendent:** Helps in managing the current active child of the Calendar control.
- **Role:** Gives information to assistive technologies about how to handle each element in a widget.
- **Grid-cell:** Defines the individual cell that can be focussed and selected.

Keyboard interaction

You can use the following keys to interact with the Calendar. This control implements keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys:

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | **Focuses the same day of the previous week.** |

| **Down Arrow** | **Focuses the same day of the next week.** |

| **Left Arrow** | **Focuses the day before.** |

| **Right Arrow** | **Focuses the next day.** |

| **Home** | **Focuses the first day of the month.** |

| **End** | **Focuses the last day of the month.** |

| **Page Up** | **Focuses the same date of the previous month.** |

| **Page Down** | **Focuses the same date of the next month.** |

| **Enter** | **Selects the currently focused date.** |

| **Shift + Page Up** | **Focuses the same date for the previous year.** |

| **Shift + Page Down** | **Focuses the same date for the next year.** |

| **Control + Upper Arrow** | **Moves to the inner level of view like month to year and year to decade.** |

| **Control + Down Arrow** | **Moves out from the depth level view like decade to year and year to month.** |

| **Control + Home** | **Focuses the first date of the current year.** |

| **Control + End** | **Focuses the last date of the current year.** |

Note: To focus the Calendar control, use **alt+t** keys.

CSHTML

```
@Html.EJS().Calendar("element").Render()
```

```
<script>
    document.addEventListener('keyup', function (e) {
        if (e.altKey && e.keyCode === 84) {
            // press alt+t to focus the control.
            var calendarObj =
document.getElementById("element").ej2_instances[0];
            calendarObj.element.querySelectorAll('.e-content
table')[0].focus();
        }
    })
</script>
```

ACCESSIBILITY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult sample()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Ensuring accessibility

The Calendar component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Calendar component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Calendar component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Displaying Islamic Calendar

In addition to the Gregorian calendar, the Calendar control supports displaying the Islamic calendar (Hijri calendar). **Islamic calendar** or **Hijri calendar** is a **lunar calendar** consisting of 12 months in a year of 354 or 355 days. To know more about Islamic calendar, refer this [wikipedia](#).

Also, it consists of all Gregorian calendar functionalities as like min and max date, week number, start day of the week, multi selection, enable RTL, start and depth view, localization, highlight and customize the specific dates.

By default, calendar mode is in **Gregorian**. You can enable the Islamic mode by setting the **calendarMode** as **Islamic**.

CSHTML

```
@Html.EJS().Calendar("calendar").RenderDayCell("customDates").CalendarMode(Syncfusion.EJ2.Calendars.CalendarType.Islamic).Render()
<script>
    var addClass = ej.base.addClass;
    function customDates(args) {
        /*Date need to be disabled*/
        if (args.date.getDate() === 12 || args.date.getDate() === 17 ||
args.date.getDate() === 22) {
            args.isDisabled = true;
        }
        /*Date need to be customized*/
        var span;
        if (args.date.getDate() === 13) {
            span = document.createElement('span');
            args.element.children[0].className += 'e-day sf-icon-cup
highlight';
            addClass([args.element], ['special', 'e-day', 'dinner']);
            args.element.setAttribute('data-val', 'Dinner !');
            args.element.appendChild(span);
        }
        if (args.date.getDate() === 23) {
            args.element.children[0].className += 'e-day sf-icon-start
highlight';
            span = document.createElement('span');
            span.setAttribute('class', 'sf-icons-star highlight');
            //use the imported method to add the multiple classes to the
given element
            addClass([args.element], ['special', 'e-day', 'holiday']);
            args.element.setAttribute('data-val', 'Holiday !');
            args.element.appendChild(span);
        }
    }
</script>
<style>
    .e-calendar .e-content td.e-selected span.e-day.sf-icon-
cup.highlight:before,
    .e-calendar .e-content td.e-selected span.e-day.sf-icon-
start.highlight:before {
        color: #ffff !important;
    }
    .sf-icon-cup:before {
        content: '\e724';
        color: #0b0bd9a1;
        position: relative;
        top: 1px;
        left: -2px;
        font-size: 11px;
        font-family: 'e-sb-icons' !important;
    }
    .sf-icon-start:before {
        content: '\e708';
        color: #0b0bd9a1;
    }
```

```

        position: relative;
        top: 1px;
        left: -1px;
        font-size: 12px;
        font-family: 'e-sb-icons' !important;
    }
    span.e-icons.highlight {
        margin-top: -13px;
        display: block;
        margin-left: 4px;
    }
    .e-other-month span.e-icons.highlight:before {
        content: "";
    }
    .e-selected span.e-icons.highlight:before {
        color: #fff;
    }
    span.e-icons.highlight,
    .e-calendar .e-content span.special,
    .e-calendar .e-content span.daycell {
        font-weight: bold;
        font-size: 14px;
    }
    .e-calendar .e-content span.special {
        border-radius: 50%;
    }
    /* custom generated icons styles */
    @@font-face {
        font-family: 'e-sb-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAAwAgTlMvMj0gSUMAAAEoAAAAVmNtYXCfQJ/XAAACSAAAAK5nbHlMkTE
7TgAAA2AAACmAAgVhZBMHhX8AAADQAAAAANmhoZWEh1QOTAAAArAAAAACRobXR4w2z/4QAAAYAAAAAD
IbG9jYdGqyAAAAAL4AAAAZmlheHABSGEWAABCAAAACBuYW1lTGtTDAAALoAAAAAJCg9zdGE/72E
AAC8sAAACyGABAAADUv9qAFoEAP/0//YD8gABAAAAAAAAAAAAAAAAAAAAAMgABAAAAAQAA3FAQWF8
PPPUACwPoAAAAANgBRFGAAAAA2AFEWp/0//8D8gPrAAAAACAACAAAAAAAAAAAAEAAAAyAQoADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPoAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnMwNS/2oAWgPrAJYAAAABAAAAAAAAABAAAAAPo//Q
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+j//gP
oAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6P/8A+gAAAPoAAAD6AAAA+g
AAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+j/9APoAAAD6AAAA+gAAAPo//8D6AAAA+gAAAPoAAAD6AA
AA+gAAAPoAAAD6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEAJAAAAKAAGAAgAC5xv
nHuc15zP//wAA5wDnHecg5yf//wAAAAAAAAAAAAEACgBAAEIATAAAAAEAAgADAAQABQAGAAcACAA
JAAoACwAMAA0ADgAPABAAEQASABMAFAAVABYAFwAZABgAGgAbABWAHQAEAB8AIAAhACIAIwAkACU
AJgAnACgAKQAQACsALAAATAC4ALwAwADEAAAAAAAAAAVgCIANABBAE+AUWBYgGMAAgBtgHOAmACsAL
UA24D2gRWBjAE0gVCBZgFuAXsBkwGegc0B5wIEgh2CTQJrgqcC1AL6gw2DJgODg6cDu4PUg/AEGQ
REBF2EfITNBnmFG4UwAAAAAP/9AAAA+oDeQASACQALwAAEWYSFXY2NxcTBRcOAScmAjcOATcRHgE
XISc3NTYnAQYiJwEOATcTFjI3ATc2NSEWBhPrxoDeTV0+/mxgQKlhorsQBw1lDB4Rabdq+QEN/ts
HFAf+2wUEMv8HEQcBAAQC/dUBAonK/tYdD15cRwGwokpMSwsXAQi3GDGR/rgSJRBZRZNoNCP7bBgY
BJQIMMf8ABwCBAAEBAoAAADAAAAAAPqA20ABAAJABKAAAKBESERJRUJATUjER4BMjEYnJURNcY
nIQ4BAfYBtvyVA2v+Sv5LPwEjGwNrGyMjG/yVGyMBdWFF/gICAHEl/rsBRYP9jxsjIxsCcRsJAQE
jAAAAAwAAAAAD6gNtABQAGQApAAABJyYOARYfARUHNtc+AS4BDwERCQE1FQkBNSMRHGEzITI2NRE
0JichDgEDrNILGBAEC/b8lekLBBAYC8UBRQG+/kL+Uz8BIxsDaxsjI xv8lRsJA TaYCAQWQGQiyKzW
oCBkWBAiOAXf+wAFLbR/+tQFBKf2PGyMjGwJxGyMBASMBQAAAAAD7QNLAQABwAKAA0AFgAANYE
lByc3BRMBJQkCISUzNzYmByIGJwYDsP5tRUWZAYED/FABff5/AdwB2PxQAv64LAYRCwCIAaLaMjI
E0gHu/hLSARz+uwFbGVgZEAECAGAAAAFAAAAAAPqA64ABQAIaAsAEQAZAAALISUXPwEFLQENARE
lASCHASUFESERNyc1JQN6/QUBJAfXQGL/u0BE/3I/s4DX/7EXl7+ugGg/gAD5wIC/hp+xgEFQOH
F/vrRaeku/ttDQAEqrI/9ZAJLAQE01AAAAAEAAAAAA+oDKwACAAA3IQECA+j+DMICaQAAAAACAAA

```

Copyright © 2001 -2024 Syncfusion Inc.

BIiYvAQEeAQcGFh8BHgEHISY2PwE+AScmNjc+ATceASUOARcWBicHDgEfASEXHgEyNj8BITc2Ji8
BBiY3NiYnLgEnDgECVQEPMD0wDwEBKioOAhk/EwM8Ew79Aw4SPQUSPhkCDiogZ0REaP5bORADFSg
HCVYSFg4BBAETVnFWewEBBA4WELUHBiWVAXA5KX9SUn6AAxshIRsDAuQ0WwTDkgQCNmUdHGy2AwO
SwQRdNCi1AQELCEV7BqN4AQZMLsgfAjDERDCChiVTAQDdqsFekUsLwEBLwAEAAAAA0cA+oAdWA
ZAB4APgAAJTIzNx4BFw4BBY4BJz4BNwEHDgEHIY4BLwELHQEHNScVHgEXMRUjDgEHHgEyNjcuASc
jNTc+ATc1NCYnIQ4BAfMCAQ1jdAEBe2loewICc2QBbgIbv4QBhb8cAQLI/TI/BNqoCWaSAwSu4q4
FBjJlCQKn2QQSDfzZDRLXAQQwGBgxAgIxGBgwBAI4B32fAgOffQacWwJdH3mq6hCZBkU+REVFRD5
FBpgBEomqeg4RAQERAAAAAIAAAAAA8YD6gBYAH8AAAEeARUOAQ8BMxYXNzY3PgEeAgYHDgEPARU
WBzEXFhceARQOAicuAS8BBgcXFhcWDgEiLgE3PgE3NTEmJzQ3NScmJy4CPgEyFx4BHwE2NzEnJic
mPgEyJw4BBxUhFR4BFy4BJzUjAREjDgEHIS4BJyMRExceATc+AScuASciAzoHBwcSBgEBBAIKHRS
ECw0IAwQGFy4XAQECCh8iBgkICw0EDxoPAQIEAQMHAgILDw4IAQcRBwcCAQUiJQYKAQgLDaQQHBA
BAwQBAwcaQwPIztJAv56EUYEKFYUoAFUAULcAQGgAVxJAFEDGDsgQUshClQ6DAPHAgoGFiwWAgE
DBxQBQCFDQ4LAWUIBQECBAMGEg4CCA4NCAEEEEicSAQIBCYQkQBwJBAoGFisXAgIIAwMBAhINAgc
PDQoEESUSAQMBBScmBgwJIALUOGMBQqCHM5cmAf5k/oQEHBQUHAQBfAEjAhMQBQpJQjtJAgAABQA
AAAAD6gPnAACAdGAmACwASwAAJSE3FxYyPwETDgEHLgEvAR4BHwEHBhYzITI2LwE3PgE3NCYjIQ4
BJQ4BIiYnJQ4BDwEjISIGFR4BFz4BPwI+ATc+ATcyNjQmIw4BAnT+iCoFQ5NCB+MQz5aXzxBAAnx
nBUIJEhIB5xIRCUIEZ3wBEg381Q4RADwQM0E0EAGkBGm3BAX+zg0SAm1NPl8TAgI9agYDhWcoERE
OdJVERQIWFgIBeJK9BAS9kh99xjkCbRAfhXbTAjnHfA0SARHfERSbEZIDOhYBEg0sWgQCQSYFARc
+BAU3BBIAeGQ+AAAAAIAAAAAA7ED6gAPAKgAACUWNj8BBw4BBY4BJzU3BhYDFBYzIxUjFR4BFxY
2PwEVDgEHLgEnFSM1IxY2Ny4BJw4BBx4BNyMVIxUeARcWNj8BBw4BBY4BJxUhNQ4BBY4BJzU3BhY
XFjY3NSM1Iz4BNS4BJw4BBxQWFyMVIzUOAQcuASc1NwYWFxY2NzUjNSMyNjcuAScOAQceATMjFSM
1IzI2NS4BJw4BBxQWMyMVIzUjMjY1LgEnDgEB7wZtKAIBA09GSU0DAgFFXhCRD4UBR0IFZSYEA0p
DP04BQhUSFgEDJQIDJQMBfXEuLRA/NgZsJwEBak1GNUMRA3YRRjZITgMBAUVQBWYmLBorFwILAwI
lAxcRDkMQRDtFSwMBAUNMBWgmhRURFwEDJQMCJQMBfHETWRcRfWILAwILAxCREVGaERcCJQMDJcg
DGkMEB0FpAwNpQQUDBlMCriMjmlESQAMDGD0IBz5kAwJSOJ6yARwiJkACAUAmIh0BsoAXMAIDGkM
CBUFpAwE+LtzIMEECA2lBBwEGUwYDFz18mwEkIyZfAgNFJyIjAZueN1MCA2Q+BwEGTwYDGUBQmyU
jJkQDA0UnIyObmyUjJkQDA0UnIyObmyUjJkQDA0UAAAAACQAAAAAD6gPhAAsAFwAjAC8APgBVAfK
AcQB7AAAlFRQWMjY9ATQmIgYHFRQWMjY9ATQmIgYlFRQWMjY9ATQmIgYFFRQWMjY9ATQmIgYnHgE
yNj8BFQ4BBY4BJzUlHgEXDgEHLgEnPgE3MxUUFjSBmjY9AScRIxEnFSMOAQcDFgQgJdCRJiQnIzU
0JisBIgY3HgEyNjcuATUGAlAJDQkJDQnxCQ0JCQ0JAdAJDQkJDQn9cAkNCQkNCVLI6f/pSAUC5s7
N5gIB96jJAWLmzs3mAgPHpwYJB10HCR8/Hwii/wwwBCQEtAX0BLQgM/v6jBAkHXQcJFAEYJBcBBSg
p5V4GCQkGXgcJCQdeBgkJB14HCQkYXQcJCQddBwkJB10HCQkHXQcJCWovMjIvBKMvWgMDWi+j8At
SLS9aAwNaLy1SC6EGCQkGoYj+5wEZEfKjYVn+8WNNz2MBDlpiCFkGCQhWEhgyEiEjJTQAaWAAAAA
DrAPqAAGAKgBgAAABHgEUBgcjNjclFgYHBjY3PgE3HgEXDgEHLgEnPgE3FAYHBjY3PgE3NjcwAw4
BFx4BFw4BBxYSFz4BNzM+ATcuAScNjcuAScuATc2JiIGBw4BFx4BFyYjIgcUATc2Jg4BAy0fKio
fQx4T/v4BFRkNkgwZGgJQWakMoZWVogsJa2AUGg0rCwxZahwgNYsyJg4IFwlxjQIazZFTjzVdNkg
BAUg2BwceAn9oFydrCAcTEwQzJg8HFgoqLCEgGCNPCAceSewGkASTBKwFGU8gaLxUNAgcVLhk0Jw0
VNgMDNhpUPKwzLhUNAgcXMxsBAQEBCyZfQB09Hw5DLO/+zwUBb2ECSjc4SgEmKCpBD1KlRgcFBQM
mXz8dPB4EA1ChRacFAQQAAAAADAAAAAPqA9UACQAVACsAAAEzHgEUBgcjNjcnDgEHLgEnPgE3HgE
FGgEXPgE3Mz4BNy4BJyM2Ny4BJw4BA0UUJDawJE0jFiUNuaqquA4OuKqqufzvHuulX6M8az5SAQF
SPggIBQXyt7fyAmIBPVs8AWJ0tB1MBARMHRxMBARMfv6w/1MHAZyJAmhPTmgCNthPaQICaQAAAAAL
/9AAAA+oD1gAcAdoAAAEGBw4BBw4BBwYHDgEHBhYXFjI3PgE3PgEnLgEjAQYCFx4BNzY3Njc+ATc
+ATc2NxcWJy4BJy4BJyIGA3lJR0B5NC1zRiIkDysMDjUTZPl/h6gVBgEVDyK/UcKpsBMGJS4oKR0
boF8l04xQWWImEQQDFwk6oF0/gwKMAyMhcEs+acQRDAQECxYgCTk+ReKBHkYWDAGBC1j+0aIxNAI
BDQoNHVU1U4MpLgMDARoOHQo/QwIeAAAIAAAAALx+oAIQBBAFAAYwCEAKIAXAEJAAALFxQOAgc
jLgM3NTY7ARYUHgIXMzI+AjUxJjSBMiUWBw4BBw4BFBYXHgE7ATIUByMiJicuATQ2Nz4BNzMyNxc
WFzMWBiMnIiYlMTQ2NxYHBisBJicxJjYeATsBNjc2MgM2FhUHBgcVFBYXFCInLgE3NTQ/ATQjJwY
PAQYmPwE2PwEWBw4BBw4BBwYjIi8BIjYzFxyZmJc+ATc+ATc2MgEhMgYWFx4BHwEeAQcGBxYGBY4
BNyYnJjY/ATY3Nj8BNDMHFA4BDwEOARceBxcUBgcOAwcOARceAjYiFj4BNzYmJy4DJy4BNzQ+BJc
2Ji8BLgI0IyEiAsABBXDNVwlcUBcIAQECAQMEF9bCVZMFQQBAwIC/vYBAw0yFagJERgoOhgPAwM
PGDsoGBYNcBUyCwEDFwIDECQDAQMkEAsHUQICCh0EGgQCBAQMDAQdBQECFQIIBwUBAwYGAQYEAQY
GASACAgEDBQIDAgSXAqMJKgsNFQ8KDBEODwMBaw8NEQwIEBUNCSOIAQP+zwFZBgEGAwMSCQcHDQQ
LIgJSdHNOAiULAwWHGwODAwICBQoGBw4VBw0DBihZMBYQCAUBCQoKI0woBgCDBglBYSACIWBBCQc
DCAUoTCMKCgQCBAGLfjBZKAYDDQcVDggFCv6nCiwBAwgJDAEBDAkIAwIBAQMFCawBDQgFAQMYAgE
EBgQBBQMIawYCBQEDBQQIDAYCAwYECQICAQEFACcAgEHAQMHAQYCAwMDAQQBAWwBBgdZRBQDjK
yAwIzOQ4QGkVYBgEBAQECAwICAgFIAQEhFgQFBgMBAQIFAQEBAwQFBBUGAQHYDUQWED0bFBNFGTQ
kAyGDayQCJjgXRBVOHhAWIioFBQDFKik+FEYYIEI1Fg8YUEIkKGkTDw8JCAMFDAUHEAUBAQUQBwU

MBQMICQ4QE2ooJENPFw4XNUQeGEUVPikqRQ8AAAAAAwAAAAAD6gPBABIAJABcAAA1HgEXIgYjDgE
HBg8BNS4BJx4BBRUuAS8BJicyNjM+ATc2Nw4BExUOAQceAQcGBw4DFBYXFhceAR8BHgEyNjc+ATc
2Nz4DNCYnJi8BNCMMnJcuASclLgEiBgMsFSCMAwcDGk0vFRcxAQgHOGX+izpmKCgUDAMHAXpNLz1
HCQoZaYiCBgpmIBsMexMMDAkcLyFXNJYPLjcuEEJ2MEIrdBMTDAwJHC8EA1QLCAKHbgEvSS79BQk
FAwgPBQUBBAYWKBEDDTgHawwHCgQFAwcQBQUFEYgClg0lpE1Fe0MJDQULEhkgGw0cDQwQBgoYHR4
aAw0JDxQEDBIYIRkJGw0DAUV9R0urIwOhLCwAAAAADAAAAAPqA+oAFAAgACwAAAErFBYXFjM2PwE
2LgEGDwERLgEiBgEOAQcuASc+ATceAQUWABc2ADcmACcGAAHXDAoFBBANgCdfRkIZgERGxEB1AX
3urn3BQX3ubr3/FsGARrU1QEaBQX+5tXU/uYDZP6TCxADAgEM0wsYEAMLiAEQDRIS/oa69wUF97q
59wUF97nV/uYFBQEaldQBGGYG/uYAAAX//wAAA+4DjQAFABEAIQAvADsAAAEVMzUjNQUOAQcuASc
+ATceAScGBx4BFzY3NiYnLgEjIgYFDgEXFjM+ATcnJgciBhMeARc+ATcuAScOAQg52Z0BSQS4iom
5Aw05iYm5KwUFQFoVDwohCCgdRB8MFv0TKAghCg8VWkAKExgiRRUF5a2t5gQE5q2t5QKs/jzCtYq
4BAS4iom4BAS49wQFL4NMAQkaaTYmKwZLNmkaCkyDLwoNASz+nq3lBATlra7lBATlAAAAAGAAAA
D6gOzAdcAQwAAQ8BBhQeAT8BFWEoARY2NwEXIw4BFBYXITY3NiYvATcfARYfARYpWE2NCcuAgY
PAQmjJyYjJhceARc+ATcuAScOAQGLA8IQHysRmUv+BxMBIi4SAVp9pRggIBgBLhgOEgMS1UtbBgU
EBxEoDsUQEAcVExUHNp0DaxIWFLYBPi4uPgEBPi4wPgNYA7sPKyEBDPNL/gYRLyECEQFegAEgMB8
BAQ8RMhTYS14GBAMDDg6/DyoSCQkDCAeWAQADEwEgLj4CAj4uLT8BAT8AAAGAAAAA8gD6wAGAA0
AFwAbACUALgBLAGYAACUHPgE1ESEBFYXIREhJSEyNj0BNCYjIQMzESMFFRQWFyE1ISIGJR4BMjY
0JiIGJQ4BBw4BBx4BFACWNjc2NzYmJz4BNzYmJyYHDgEFHgEXFAYXFh8BFjYXFjY3JjY3NiYnLgE
nJgYCMgFIDhT+lv4fFA4BSP6WAeEBdA4UFA7+jGdUVP5aFA4BdP6MDhQBdAE0UTQ0UTQB1SNEIhg
mBwwQCRaVGGziJyQGAQ4EAgcOCQ0TKf0DBg8BCw4LEyAhQyMKUAEMBhUDPgoPjTsaGQIBew8B9/4
JDxMBAhopFA5hDhT9GALoImEOFQKoFGwfKCG+KCh0ChwMBxYSchocDgsCBRMKCTsTDhkOCA8FBAE
CDBQMGGwWLREJBAYGAQEBBakYKRAHJAQIKwIBEAJAAAAA0sA+sABQAJAA8AEwAYABwALAA8AHI
AAAEsMxEjESMRiXejOwERiXelFSM1ITMVITUjFSM1JR4BDgEPASM3PgE3NjczFiUWFx4BHWerAS4
CNjc2MyMOARcWHwEjIgYHFR4BFxEUFhchPgE3ET4BNzUuASsBNTY3NiYnJgciIwYHBg8BJyYnJi8
BJgKyfbs/fX4BP7wCsJ3+88/+xz6cAkIDCGZBWggJAQcWCxoZBRz+2RkaCxYHAQYCXkUGCQIPHHM
PDgg0OwGwGyMBASMBIxscCsRsjARojAQEjGrA7DgoXCj03BgU5KQwKAwQKCyk5CzcB9v5LAbX+SwG
1/ksBtbx9fX19fX3SBBcgNx8DBiNKfi8EAQEELxZJIwYgOCIVAiorNB8wJwEjG30bIwH+SxsjAQE
jGwG1ASMBfRsjAScvJzYJPgEISxchCQwfFksIAQEAAAGAAAAA+oD6gAGAA4AFgAdACQAKgAyAdo
AACU+ATcjHgEFPPgE3LgEnIwEeARcRIw4BJz4BNyMeAQmZLgEnDgEFMyYnDgEHMz4BNy4BJwUeARc
zEQ4BA4YrNAXnBkv+wlygP0JUBp/+f0CkXqwfUHMxRgbeBTI33ghFMCoyAvznCloyS/OfCFJCPaJ
c/oI+UAWsXqPHOYpNaI/eBkk7KbBy/rG9SgYB1W+tDxuMZkuHARFmjBs7h0uadhmpaHkWKtTJBo0
srW8B1QZKAAQAAp//A8YD6gADAC4ANwBOAAABMZUjJQYWHwEVAwYWFzMWnJcTFxYXmYjY3Ex4BNzM
+AScDNTMyNjQmJyEnJiMiBjceATI2LgEiBiUUFzYWFxYGDwEXPgE3NiYvASyJjW4BAw64uP0hCwg
TanoGDxQDEyAIYQMKDwcNBWEHIRMDEXAGDP0UICAU/nNqCg8MFXgBMEgxAi9IMAFwCQdqEwcPbA8
fCYMHBBy6BgMEBgoMAokZRREjCki//qITIQCHEBMBGQMFAQIE/ucTEAchIRMBsMsWJhsBSAKMLiQ
vL0gwMI8LBQGHkAfXgBAiAriaCel7BwIBAw0ADgAAAAAD6wPqAAIABgAWABkALgBkAGgAbAB/AJs
AqQCvALQAZwAAJTC1Nwc3BicOAQczPgE/AQc2Jy4BLwE/ATUnBg8BMx4BFxU3Nj8BNDUnLgEnJic
lBg8BFQYVfHIXMyC+AT8BIYjYDgEHDgEjBgQvARQ/ATY3PgE3PgE3Nh8BNz4BNzUnJiQHDgEnBhU
3MT8BNSUOAQcVHgEdAR4BHWE3Nic0LwEBDgEHFT4BNzYEHwE1Nic1LgEnLgE1LgEnJiMnARc+Aj8
BJy4BLwEeASUGDwI3BSYvASMFBzc2FhcyFhCWHEWHwE1LgEvASyZiY4BIwYDfQMBaQYDTiarmSt
vv0oDEwQBARYOBKYDgwUOAwwMWJwIGMBgDEwkbDiEX/QImFxoGBPzFAz+cuyoDaxwmM2AZBwYCPf7
6CBkmLhshQptMA31PKSUGAwgMAgZx/vN8QGx+AgMBAgOzHD4NagQEVSdAwQBIGp9WmSMHQebgXo
BDnUGAQQCnRYcIkfBPQkGFwIpAwsoHwkDAyh+TgYmb/4RBQYLAyMBZw4QHQT+5AYfCM1qAgcGCAo
XISEDJHgdBAECBho6HVO+AwECAGgDhFkKuwFXTgMTIRgxPBADMGwKOyoqBhNRRQMJRlMKAgESchI
KFw53BQMEAYymx/7wEwlvrlUGCGEXCwIETysDJgEEBQQHcy4nAyYDAQ0EBxUnFQYDPI4BAQCEBQC
QAgcGQiMhBAMOHw8iAjUgAwwdIWRVBgEDL6pvAwQRAQIqPAMNIBSGCCMUGCUBUYWEQAgH+3QkJEx4
RAwZLCiEDBpukAgMEAw8DBQKQKQKBFGLCACkChUYFAMDD6UaCAQHBgEAAAABAAAAAAPPa+oAGgA
AASBs8g8BJQcFBycHHWE3JzctNwM3NjclNic3A3wKaw4O4f3/VAGsrIZFlk5CEKzbV4PkDgUEARM
D2AU0BZBa7rgSSFSiSiY4/j5eAhr0DgoECARUAAAABQAAAAAD6gPJABsANgBMAKQyARQAjQ4BBY4
BJzY/ARceARceAT4BLwEmLwE3NjMeASUXIw4BIw4BHgEXNXy2NxcOAQcuASc+ATcyFhMeARcWHwE
HDgEHFScuAicmNjcwFycOAQcOARceAR8BIwYHIYcuASMOAQceARc+ATc0LwE3Nj8BFQ4BBwYWFzM
yNjc+AS8BNTY3Fh8BBw4BBx4BFz4BNy4BJyIHIYcuATc0JicjJyYvAS4BIwY3HgEyNjQmIgYDrAJ
HNTVHAgerAwQbJAMKGRMBRCRQSGwQEGb41R/2BAQExQgUNEAMRDAMDSdGBakc1NUcCAkc1IzhyGB0
QCyRIAgwwIAEXRUKMDCIUCgwqCRcQFB4TE1cvHQElJwECGvc2UGoDA2pQUGsCAQEHKSgCAQYFARA
NBawRagYHBAE0IxYfBgUUFgECa1BQawICa1AxKQIDHSGCDQsCkwYJERwvFwgGASM2IyM2I+e2RwE
BRzYkhQGGJCSDCQESGQoXFiYEAgwBRwWDEAwCExoPAQEABDRMDNkcBAUC2NUcBIQFVIYABAQECDDd
VIAEBCxskFSWMKQwQYwMdJi+SMYyWegwbEgMtNgJqUFBrAgJrUAKKCAQSHAEFI1wuDRMCDw0+fSg

```
FATBIPDQKBRk9tI1BrAgJrUFBqAhcGMXE5CxIDbQCmFyo1AXYbIyM2IyMAAAAAABwAAAAAADwPpqAAAM
ABgAKAAwAEAAUAC4AAAERIRElBzUnFSElISMnISc3ASMBNycBBhQfASmiBhURFBYzITI2NzU3NjQ
nASYiAxz9NANNQj/9NALrH+7+hES/Aeln/rwzLP7MEhIYChokJBocZBsJA4Sev4GEzMBXP7lARt
WQoQCPDw+RL/+ /QFFNCz+yxMyExgkGv5LGiQkGtZvEzITAFsSAAAAABIA3gABAAAAAAAAAAAAEAAs
AAAAAAAAABAaOAAQABAAAAAAAAACAaCwABAAAAAAAAADAAoAEgABAAAAAAAAEEAAoAHAABAAAAAAFAAs
AJgABAAAAAGAAoAMQABAAAAAAKACwAOwABAAAAAALABIAZwADAEEECQAAAAIAeQADAAEEECQA
BABQAewADAAEEECQACAA4AjwADAEEECQADABQAnQADAAEEECQAEABQAsQADAAEEECQAFABYaxQADAAE
ECQAGABQA2wADAEEECQAKAfG7wADAEEECQALACQBRYBlLXNiLWLjb25zUmVndWxhcmUtc2ItaWN
vbnnLlLXNiLWLjb25zVmVyc2lubiAxLjBlLXNiLWLjb25zMm9udCBnZW5lcmlF0ZWQgdXNpbmcgU3l
uY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABlAC0AcwBiAC0AaQBjAG8
AbgBzAFIAZQBnAHUAAbABhAHIAZQAtAHMAYgAtAGkAYwBvAG4AcwBlAC0AcwBiAC0AaQBjAG8AbgB
zAFYAZQByAHMAAQBVvAG4AIAAxAAC4AMABlAC0AcwBiAC0AaQBjAG8AbgBzAEYAwbBuAHQAIAbnAGU
AbgBlAHIAIAYQB0AGUAZAAGAHUAacwBpAG4AZwAgAFMAeQBuaGMAZgBlAHMAaQBVvAG4AIAbnAGUAdAB
yAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgBlAHMAaQBVvAG4ALgBjAG8AbQAAAAA
CAAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAADIBAgEDAQQBBQEGAQcBCAEJAQoBCwEMAQ0
BDgEPARABEQESARMBFAEVARYBFWEYARKBGgEbARWBHQEeAR8BIAEHASIBIWekASUBJgEnASgBKQE
qASSBLAETAS4BLwEWATEBMGeZAAltYWlsLXNlbmQLaW5ib3gtMDItZDYLaW5ib3gtMDEtd2YFaW5
ib3gPb3Blbi1tZXNzYwdlLXdmdGFycm93aGVhZC0wMRBhcnJvd2hlYWQtdG9wLXdmcXJhdGluzY1
3ZgtYXRpbmctLS0wMxPhcnJvd2hlYWQtdG93bi0wMQlmb2xkZXItMDMIdXNlcnMtd2YIY2xvY2s
tMDIGdXBsb2FkCG9uZWRYaXZleWNSb3VkLWRvd25sb2FkLXdmd3dvcmstb2ZmbGluZS13ZglzZWFW
yY2gtZDYpbn90ZS1tZW1vLTaxLXdmdGltcG9ydGFudC13ZglzZW50LW1haWwIaw5ib3gtZDYLaWF
pbC0tLXNlbnQJdXNlcilYWNrDHVzZXItcHJvZmlsZQ5hbGFybS1jbG9jay13Zg5zZWFWyY2gtZG1
tZS13ZgtiZWsLW5ldy13ZhdiZXZlcmFnZS1jb2NrdGFpbC0wNCC13ZhRiZXZlcmFnZS1jb2NrdGF
pbC0wMGdib3dsLXdmb2Nha2UtMDIHYZ2FrZS13Zgtjb2ZmZWUtdLS0wMQ9jb2ZmZWUtdY3VwLS0tMDI
LY29mZmVlLWljYw4Nd2luZS1nbGFzczy0wMQRIzWxsDGFSYXJtLTaxLXdmdLQlhbGFybS0tMDIHXYR
obGV0ZQdnaWZ0LTaxB2dpZnQtZDYJYmFsbC0tLTazB2FyY2hlcnkKYmFza2V0YmFsbAlhZXJvcGx
hbmUNYmljeWNsZS0wMy13ZghdG0tMDItZDYAAAAA format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"],
[class*=" sf-icon-"] {
    speak: none;
    font-size: 15px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
/* end of custom generated icons styles */
</style>
```

ISLAMIC-MODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2Sample.Controllers
{
    public class CalendarController : Controller
    {
```

```
public ActionResult sample()  
{  
    return View();  
}  
}
```

Note: [View Sample in GitHub.](#)

Style and appearance in Calendar Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the background color for the Calendar

Use the following CSS to customize the background color and border for the Calendar.

```
`css  
  
/ To specify background color and border /  
  
.e-calendar {  
    background-color: peachpuff;  
    border: 3px solid red;  
}
```

Customizing the Calendar date elements on hovering

Use the following CSS to customize the date elements on hovering in the Calendar.

```
`css  
  
/ To specify background color, color, and border /  
  
.e-calendar .e-content td:hover span.e-day, .e-calendar .e-content td:focus span.e-day, .e-bigger.e-small  
.e-calendar .e-content td:hover span.e-day, .e-bigger.e-small .e-calendar .e-content td:focus span.e-day  
{  
    background-color: red;  
    border: 2px solid;  
    color: #212529;  
}
```

Customizing the border of date cell grid

Use the following CSS to add the border to the date cell grid.

```
`css  
  
/ To specify border /  
  
.e-calendar .e-content span.e-day, .e-bigger.e-small .e-calendar .e-content span.e-day {
```

```
border: 1px solid;  
}
```

,

Customizing the Calendar title

Use the following CSS to customize the Calendar title.

```
`css
```

```
/ To specify color and font size /
```

```
.e-calendar .e-header .e-title, .e-bigger.e-small .e-calendar .e-header .e-title {
```

```
color: black;
```

```
font-size: 20px;
```

```
}
```

,

Customizing the previous and next icon

Use the following CSS to customize the previous and next icon.

```
`css
```

```
/ To specify color and border /
```

```
.e-calendar .e-header span, .e-bigger.e-small .e-calendar .e-header span {
```

```
border: 1px solid;
```

```
color: chocolate;
```

```
}
```

,

Customizing the footer button

Use the following CSS to customize the footer button.

```
`css
```

```
/ To specify background color, color, and border-color /
```

```
.e-calendar .e-btn.e-today.e-flat.e-primary, .e-calendar .e-css.e-btn.e-today.e-flat.e-primary {
```

```
background-color: red;
```

```
border-color: black;
```

```
color: black;
```

```
}
```

,

Customizing the selected date cell grid

Use the following CSS to customize the selected date cell grid in Calendar.

```
`css
```

/ To specify background color and color /

```
.e-calendar .e-content td.e-focused-date.e-today span.e-day {
background-color: maroon;
color: #fff;
}
`
```

Customizing the content header in Calendar

Use the following CSS to customize the content header in Calendar.

`css

/ To specify background /

```
.e-calendar .e-content thead, .e-bigger.e-small .e-calendar .e-content thead {
background: aquamarine;
}
`
```

How To

Set clear button in Calendar Control

To configure **clear** button in Calendar UI, do the following:

1. To the [created](#) event of the Calendar, add the required elements to make clear button visible. In the following example, div with Essential JS 2 button control is used.
2. When the **e-footer** class is used, the div tag acts as the footer.
3. Using these button, selected date can be cleared.
4. Bind the required event handler to the button tag to update the value.

CSHTML

```
@Html.EJS().Calendar("calendar").Created("onCreate").Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        document.querySelector('.e-footer-container .e-
clear').addEventListener('click', function () {
            calendarObject =
document.getElementById('calendar').ej2_instances[0];
            calendarObject.value = null;
            calendarObject.dataBind();
        });
    });
    function onCreate() {
        //creates the custom element for clear button.
        var clearBtn = document.createElement('button');
        var footerElement = document.getElementsByClassName('e-footer-
container')[0];
        clearBtn.className = 'e-btn e-clear e-flat';
        clearBtn.textContent = 'Clear';
        footerElement.prepend(clearBtn);
    }
}
```

```

        this.element.appendChild(footerElement);
    }
</script>
<style>
    .e-clear { /* csslint allow: adjoining-classes*/
        margin-right: 81px;
    }
</style>

```

SETTODAY.CS

Note: [View Sample in GitHub.](#)

Show dates of other months

The following example demonstrates how to show dates of other months.

Using the following styles, you can bring the dates of other months to visibility from its hidden state.

```

`css
.e-calendar .e-content tr.e-month-hide,
.e-calendar .e-content td.e-other-month>span.e-day {
display: block;
}
.e-calendar .e-content td.e-month-hide,
.e-calendar .e-content td.e-other-month {
pointer-events: auto;
touch-action: auto;
}
`

```

CSHTML

```

@Html.EJS().Calendar("element").Render()
<style>
    .e-calendar .e-content tr.e-month-hide, /* csslint allow: adjoining-
classes*/
    .e-calendar .e-content td.e-other-month > span.e-day { /* csslint allow:
adjoining-classes*/
        display: block;
    }
    .e-calendar .e-content td.e-month-hide, /* csslint allow: adjoining-
classes*/
    .e-calendar .e-content td.e-other-month { /* csslint allow: adjoining-
classes*/
        pointer-events: auto;
        touch-action: auto;
    }
</style>

```

```
}
</style>
```

SHOWDATES.CS

Note: [View Sample in GitHub.](#)

Select a sequence of dates in Calendar

The following example demonstrates how to select the week dates of chosen date in the Calendar using [values](#) property, when [isMultiSelection](#) property is enabled. Methods of Moment.js is used in this sample for calculating the start and end of week from the selected date.

CSHTML

```
@Html.EJS().Calendar("calendar").Change("onChange").IsMultiSelection(true).Render()
<div class='e-btn-group e-vertical'>
    @Html.EJS().Button("workweek").Content("Work Week").Render()
    @Html.EJS().Button("week").Content("Week").Render()
</div>
<script>
    function onChange(args) {
        calendar = document.getElementById('calendar').ej2_instances[0];
        var startOfWeek = moment(calendar.value).startOf('week');
        var endOfWeek = moment(calendar.value).endOf('week');
        if (calendar.element.classList.contains('workweek')) {
            getWeekArray(startOfWeek.day(1), endOfWeek.day(5));
        }
        else if (calendar.element.classList.contains("week")) {
            getWeekArray(startOfWeek, endOfWeek);
        }
    }
    function getWeekArray(startOfWeek, endOfWeek) {
        var days = [];
        var day = startOfWeek;
        while (day <= endOfWeek) {
            days.push(day.toDate());
            day = day.clone().add(1, 'd');
        }
        calendar.values = days;
    }
    document.addEventListener('DOMContentLoaded', function () {
        var calendar = document.getElementById('calendar').ej2_instances[0];
        /*selected current week dates when click the button*/
        document.getElementById('workweek').addEventListener('click',
function () {
    if (calendar.element.classList.contains('week')) {
        calendar.element.classList.remove('week');
    }
    calendar.element.classList.add('workweek');
})
        /*selected current week dates when click the button*/
```

```

        document.getElementById('week').addEventListener('click', function
    () {
        if (calendar.element.classList.contains('workweek')) {
            calendar.element.classList.remove('workweek');
        }
        calendar.element.classList.add('week');
    });
});
</script>
<style>
    .btncontainer {
        display: inline-block;
        float: right;
        margin-left: 85px;
        margin-top: 120px;
    }
    .e-btn-group.e-vertical{
        margin-top: 25px;
    }
</style>

```

MULTI.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
// For more information on enabling MVC for empty projects, visit
// https://go.microsoft.com/fwlink/?LinkID=397860
namespace EJ2CoreSampleBrowser.Controllers
{
    public class CalendarController : Controller
    {
        public ActionResult Multi()
        {
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

Skip a month in the Calendar

The following example demonstrates how to skip a month in the Calendar while clicking the previous and next icons. In the example below, the [navigated](#) event is used to skip a month with [navigateTo](#) method.

CSHTML

```

@Html.EJS().Calendar("element").Navigated("onNavigate").Render()
<script>
    function onNavigate(args) {
        var date;
        if ((args.event.currentTarget).classList.contains('e-next')) {

```



```

        //Increments the month while clicking the next icon.
        date = new Date(args.date.setMonth(args.date.getMonth() + 1));
    }
    if ((args.event.currentTarget).classList.contains('e-prev')) {
        //Decrements the month while clicking the previous icon.
        date = new Date(args.date.setMonth(args.date.getMonth() - 1));
    }
    if (args.view == 'Month') {
        calendarObject =
document.getElementById('element').ej2_instances[0];
        calendarObject.navigateTo('Month', date);
    }
}
</script>

```

SKIPMONTH.CS

Note: [View Sample in GitHub.](#)

Render the Calendar with week numbers

You can enable `weekNumbers` in the Calendar by using the [weekNumber](#) property.

CSHTML

```
@Html.EJS().Calendar("element").WeekNumber(true).Render()
```

WEEKNUMBER.CS

Note: [View Sample in GitHub.](#)

Change the first day of the week

The Calendar provides an option to change the first day of the week by using the [firstDayOfWeek](#) property. Generally, the day of the week starts from 0 (Sunday) and ends with 6 (Saturday).

Note: By default, the first day of the week is culture specific.

The following example shows the Calendar with `Tuesday` as the first day of the week.

CSHTML

```
@Html.EJS().Calendar("element").FirstDayOfWeek(2).Render()
```

FIRSTDAY.CS

Note: [View Sample in GitHub.](#)

Customize the calendar day header

You can change the format of the day that has to be displayed in header using [dayHeaderFormat](#) property.

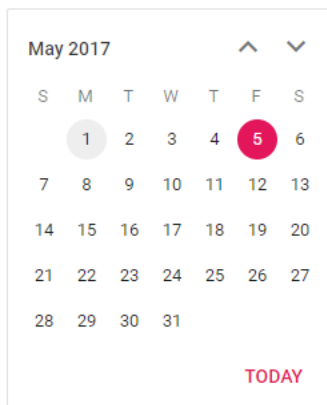
Note: By default, the format is **Short**.

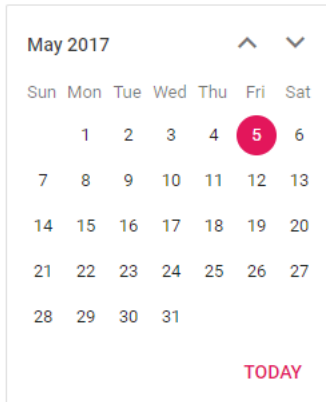
Name	Description
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

CSHTML

```
@Html.EJS().Calendar("element").DayHeaderFormat(DayHeaderFormat.Short).Render()
```

HEADERFORMAT.CS





Note: [View Sample in GitHub.](#)

Card

Getting Started with ASP.NET MVC Card Control

This section briefly explains about how to include [ASP.NET MVC Card](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Add stylesheet

Here, the theme is referred using CDN inside the `<head>` of `~/Views/Shared/_Layout.cshtml` file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls.

Add ASP.NET MVC Card control

Now, add the Syncfusion ASP.NET MVC Card control in `~/Home/Index.cshtml` page.

CSHTML

```
<div class = "e-card">Sample Card</div>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Card control will be rendered in the default web browser.

Adding a header and content

You can create Card with a header in a specific structure. For adding header you need to create a `div` element with `e-card-header` class added.

- You can include heading inside the Card header by adding a `div` element with `e-card-header-caption` class, and also content will be added by adding element with `e-card-content`. For detailed information, refer to the [Header and Content](#).

```
`html
```

```
<div class = "e-card">           --> Root Element
<div class="e-card-header">      --> Root Header Element
<div class="e-card-header-caption"> --> Root Heading Element
<div class="e-card-header-title"></div> --> Heading Title Element
</div>
<div class="e-card-content"></div> --> Card content Element
</div>
</div>
`
```

CSHTML

```
<div tabindex="0" class="e-card" id="basic">
  <div class="e-card-header">
    <div class="e-card-header-caption">
      <div class="e-card-header-title">Advanced UWP</div>
    </div>
  </div>
  <div class="e-card-content">
    Advanced UWP: Communicating with Windows 10 and Other Apps, the
    second in a five-part series written by Succinctly series
    author Matteo Pagani. To download the complete white paper, and
    other papers in the series, visit
    the White Paper section of Syncfusion's Technology Resource Portal.
  </div>
</div>
```

Advanced UWP

Advanced UWP: Communicating with Windows 10 and Other Apps, the second in a five-part series written by Succinctly series author Matteo Pagani. To download the complete white paper, and other papers in the series, visit the White Paper section of Syncfusion's Technology Resource Portal.

Note: [View Sample in GitHub.](#)

See also

- [Real time example using Card](#)
- [How to add a header and content](#)

Header and Content in Card Control

Header

The Card can be created with header title, sub title and images. For adding header, you need to create `div` element with the class `e-card-header` added.

Card provides below elements and corresponding class definitions to include header.

Elements | Description

`e-card-header-caption` | To group the title and subtitle within the header which acts as wrapper.

`e-card-header-title` | Main title text within the header.

`e-card-sub-title` | A sub-title within the header.

`e-card-header-image` | To include heading image within the header.

`e-card-corner` | To add rounded corner for the image.

Title and Subtitle

For adding header to the Card, you need to create wrapper `div` element with `e-card-header-caption` class.

- Place the `div` element with `e-card-header-title` class inside the header caption for adding main title.
- Place the `div` element with `e-card-sub-title` class inside the header caption element for adding sub-title.

Image

Card header has an option for adding images in the header. It is aligned with either before or after the header based on the HTML element positioned in the header structure.

- The header image can be added by creating a `div` element with `e-card-header-image` class which can be placed before or after the header caption wrapper element.

CSHTML

```
<div style="margin: 50px;">
```

```

<div tabindex="0" class="e-card">
  <div class="e-card-header">
    <div class="e-card-header-image football"></div>
    <div class="e-card-header-caption">
      <div class="e-card-header-title"> Laura Callahan</div>
      <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
    </div>
  </div>
</div>
<div style="margin-left: 50px;margin-top:30px">
  <div tabindex="0" class="e-card">
    <div class="e-card-header e-card-corner">
      <div class="e-card-header-caption">
        <div class="e-card-header-title"> Laura Callahan</div>
        <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
      </div>
      <div class="e-card-header-image football"></div>
    </div>
  </div>
</div>
<style>
  .e-card .e-card-header .e-card-header-image.football {
    background-image: url('./football.png');
  }
  .e-card {
    width: 300px
  }
</style>

```

CONTROLLER.CS

```

public ActionResult Index()
{
    return View();
}

```

Content

Content in Card holds texts, images, links and all possible HTML elements. It's adaptable within the Card root element.

- Create a `div` element with the class `e-card-content`.
- Place content `div` element in the Card root element or within any Card inner elements.

CSHTML

```

<!--element which is going to render the Card-->
<div tabindex="0" class="e-card">
  <div class="e-card-header">
    <div class="e-card-header-image football"></div>
    <div class="e-card-header-caption">
      <div class="e-card-header-title"> Laura Callahan</div>

```

```

        <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
    </div>
</div>
<div class="e-card-content">
    Laura received a BA in psychology from the University of Washington.
    She has also completed a course in business French. She reads and writes
    French.
</div>
</div>
<style>
    .e-card .e-card-header .e-card-header-image.football {
        background-image: url('../football.png');
    }
    .e-card {
        width: 300px
    }
</style>

```

CONTROLLER.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Images and Divider in Card Control

Images

The Card supports to include images within the elements, you can add image as direct element anywhere inside card root by adding the `e-card-image` class to `div` element. Using the class defined, you can write CSS styles to load images to that element.

Note: By default, card images occupies full width of its parent element.

```
`html
```

```

<div class = "e-card">
<div class="e-card-image">

</div>
</div>
`

```

Title

Card image is supported to include a title or caption for the image. By default, Title is placed over the image on left-bottom position with overlay.

```
`html
```

```

<div class = "e-card">
<div class="e-card-image">

```

```
<div class="e-card-title"></div>
```

```
</div>
```

```
</div>
```

```
,
```

CSHTML

```
<!--element which is going to render the Card-->
<div class="e-card">
    <div class="e-card-image">
        <div class="e-card-title">JavaScript </div>
    </div>
    <div class="e-card-content"> JavaScript Succinctly was written to give
    readers an accurate, concise examination of JavaScript objects and their
    supporting nuances, such as complex values, primitive values, scope,
    inheritance, the head object, and more. </div>
</div>
<style>
    .e-card-image {
        background: url('../sample.jpg');
        height: 160px;
        ;
    }
    .e-card {
        width: 300px;
        margin: auto;
    }
</style>
```

CONTROLLER.CS

```
public ActionResult Index()
{
    return View();
}
```




Divider

Divider used to separate the elements inside the card. You can add divider inside the card elements to separate it.

- Place the `div` element with `e-card-separator` class inside the card element for adding a divider.

CSHTML

```
<div style="margin: 50px;">
  <div tabindex="0" class="e-card" id="basic">
    <div class="e-card-title">Explore Cities</div>
    <div class="e-card-separator"></div>
    <div class="e-card-content">
      Sydney is a city on the east coast of Australia. Sydney is the
      capital city of New South Wales. About four million people
      live in Sydney which makes it the biggest city in Oceania.
    </div>
    <div class="e-card-separator"></div>
    <div class="e-card-content">
      New York City has been described as the cultural, financial, and
      media capital of the world, and exerts a significant impact
      upon commerce and etc.,
    </div>
    <div class="e-card-separator"></div>
    <div class="e-card-content">
      Malaysia is one of the Southeast Asian countries, on a peninsula
      of the Asian continent, to a certain extent; it can be recognized
      as part of the Asian continent.
    </div>
  </div>
</div>
```

CONTROLLER.CS

```
public ActionResult Index()  
{  
    return View();  
}
```

Explore Cities

Sydney is a city on the east coast of Australia. Sydney is the capital city of New South Wales. About four million people live in Sydney which makes it the biggest city in Oceania.

New York City has been described as the cultural, financial, and media capital of the world, and exerts a significant impact upon commerce and etc.,

Malaysia is one of the Southeast Asian countries, on a peninsula of the Asian continent, to a certain extent; it can be recognized as part of the Asian continent.

Note: [View Sample in GitHub.](#)

See also

- [How to customize the card image title position](#)

Action Buttons in Card Control

You can include Action buttons within the Card and customize them. Action button is a `div` element with `e-card-actions` class followed by button tag or anchor tag within the card root element.

- For adding action buttons you can create button or anchor tag with `e-card-btn` class within the card action element.

```
`html  
<div class = "e-card">  
  <div class="e-card-actions">  
    <button class="e-card-btn"></button>  
    <a href="#"></a>  
  </div>  
</div>  
,
```

Vertical

By default, action buttons positioned in horizontal alignment, and also it can be aligned to show in vertical alignment by adding `e-card-vertical` class.

```
`html
<div class = "e-card">
  <div class="e-card-actions e-card-vertical">
    <button class="e-card-btn">More</button>
    <a href="#">Share</a>
  </div>
</div>
`
```

CSHTML

```
<!--element which is going to render the Card-->
<div class="e-card" style="max-width:400px">
  <div class="e-card-header-title">Eiffel Tower</div>
  <div class="e-card-content">
    The Eiffel Tower is acknowledged as the universal symbol of Paris and
    France.
  </div>
  <div class="e-card-actions e-card-vertical">
    <button class="e-card-btn">LIKE</button>
    <button class="e-card-btn">SHARE</button>
  </div>
</div>
```

CONTROLLER.CS

```
public ActionResult Index()
{
    return View();
}
```

Eiffel Tower

The Eiffel Tower is acknowledged as the universal symbol of Paris and France.

LIKE

SHARE

Note: [View Sample in GitHub.](#)

See also

- [How to integrate other component inside the card](#)

Horizontal Card in Card Control

By default, all the card elements are aligned vertically one after the other as in the DOM. You can achieve the element to align horizontally as well as by adding the class `e-card-horizontal` in the root card element.

Stacked cards

- An horizontally aligned card can push a specific column to align vertical using `e-card-stacked` class. This will align the stacked section vertically aligned differentiating from horizontal layout.

Class | Description

`e-card-horizontal` | To align card elements horizontally.

`e-card-stacked` | To align elements vertically within the horizontal layout.

`html

```
<div tabindex="0" class="e-card e-card-horizontal">
 --> Aligned in horizontal
<div class="e-card-stacked">    --> Aligned in horizontal
// Inside the element all are aligned vertical directions
</div>
</div>
`
```

CSHTML

```
<!--element which is going to render the Card-->
<div tabindex="0" class="e-card e-card-horizontal" style="width:400px">
  
  <div class="e-card-stacked">
    <div class="e-card-header">
      <div class="e-card-header-caption">
        <div class="e-card-header-title">Philips Trimmer</div>
      </div>
    </div>
    <div class="e-card-content">
      Powered by the innovative DuraPower Technology which optimizes
      power consumption, Philips trimmers are designed to last longer
      than 4 ordinary trimmers.
    </div>
  </div>
</div>
```

CONTROLLER.CS

```
public ActionResult Index()  
{  
    return View();  
}
```



Philips Trimmer

Powered by the innovative DuraPower Technology which optimizes power consumption, Philips trimmers are designed to last longer than 4 ordinary trimmers.

Note: [View Sample in GitHub.](#)

Carousel

Getting Started with ASP.NET MVC Carousel Control

This section briefly explains about how to include [ASP.NET MVC Carousel](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Carousel control

Now, add the Syncfusion ASP.NET MVC Carousel control in **~/Home/Index.cshtml** page.

CSHTML

```
<div class="col-lg-12 control-section default-carousel-section">
<div class="e-sample-resize-container carousel-sample">
@ (Html.EJS().Carousel("defaultCarousel").CssClass("default-carousel")
.Items(new List<CarouselItem> {
new CarouselItem { Template = "#itemTemplate1" },
```

```

new CarouselItem { Template = "#itemTemplate2" },
new CarouselItem { Template = "#itemTemplate3" },
new CarouselItem { Template = "#itemTemplate4" },
new CarouselItem { Template = "#itemTemplate5" }
})
.Render()
)
</div>
</div>
<script id="itemTemplate1" type="text/x-template">
<figure class="img-container">

<figcaption class="img-caption">Golden Gate Bridge, San
Francisco</figcaption>
</figure>
</script>
<script id="itemTemplate2" type="text/x-template">
<figure class="img-container">

<figcaption class="img-caption">Spring Flower Trees</figcaption>
</figure>
</script>
<script id="itemTemplate3" type="text/x-template">
<figure class="img-container">

<figcaption class="img-caption">Oddadalen Waterfalls, Norway</figcaption>
</figure>
</script>
<script id="itemTemplate4" type="text/x-template">
<figure class="img-container">

<figcaption class="img-caption">Anse Source d'Argent,
Seychelles</figcaption>
</figure>
</script>
<script id="itemTemplate5" type="text/x-template">
<figure class="img-container">

<figcaption class="img-caption">Stonehenge, England</figcaption>
</figure>
</script>
<style>
.default-carousel-section .carousel-sample {
margin: 0 auto 2em;
max-width: 500px;
height: 300px;
}
.default-carousel .e-carousel-items .e-carousel-item .img-container {
height: 100%;
}
.default-carousel .e-carousel-items .e-carousel-item .img-caption {
bottom: 4em;

```

```
color: #fff;  
font-size: 12pt;  
height: 2em;  
position: relative;  
padding: 0.3em 1em;  
text-align: center;  
width: 100%;  
}  
</style>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Carousel control will be rendered in the default web browser.



Note: [View Sample in GitHub.](#)

Populating Items in ASP.NET MVC Carousel control

In the Carousel, slides can be rendered in two ways as follows,

- Populating items using carousel item
- Populating items using data source

Populating items using carousel item

When rendering the Carousel component using items binding, you can assign templates for each item separately or assign a common template to each item. You can also customize the slide transition interval for each item separately. The following example code depicts the functionality as item property binding.

CSHTML

```
@using Syncfusion.EJ2.Navigations;  
<div class="container">  
    <div class="control-container">
```



```

        @ (Html.EJS().Carousel("defaultCarousel").Items(new
List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
        })
        .Render()
    )
</div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```

Populating items using data source

When rendering the Carousel component using data binding, you can assign a common template only for all items using the [ItemTemplate](#) property. You cannot set the interval for each item. The following example code depicts the functionality as data binding.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").DataSource((IEnumerable<object>)ViewBag.dataSource).ItemTemplate("<div class='slide-
content'>${Title}</div>").Render())
    </div>

```

```
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public IActionResult Index()
{
    List<CarouselDataBinding> datasrc = new
List<CarouselDataBinding>();
    datasrc.Add(new CarouselDataBinding
    {
        Id = 1,
        Title = "Slide 1",
    });
    datasrc.Add(new CarouselDataBinding
    {
        Id = 2,
        Title = "Slide 2",
    });
    datasrc.Add(new CarouselDataBinding
    {
        Id = 3,
        Title = "Slide 3",
    });
    datasrc.Add(new CarouselDataBinding
    {
        Id = 4,
        Title = "Slide 4",
    });
    datasrc.Add(new CarouselDataBinding
    {
        Id = 5,
        Title = "Slide 5",
    });
    ViewBag.dataSource = datasrc;
    return View();
}

public class CarouselDataBinding
{
    public int Id { get; set; }
    public string Title { get; set; }
}
```

Selection

The Carousel items will be populated from the first index of the Carousel items and can be customized using the following ways,

- Select an item using the property.
- Select an item using the method.

Select an item using the property

Using the [SelectedIndex](#) property of the Carousel component, you can set the slide to be populated at the time of initial rendering else you can switch to the particular slide item.

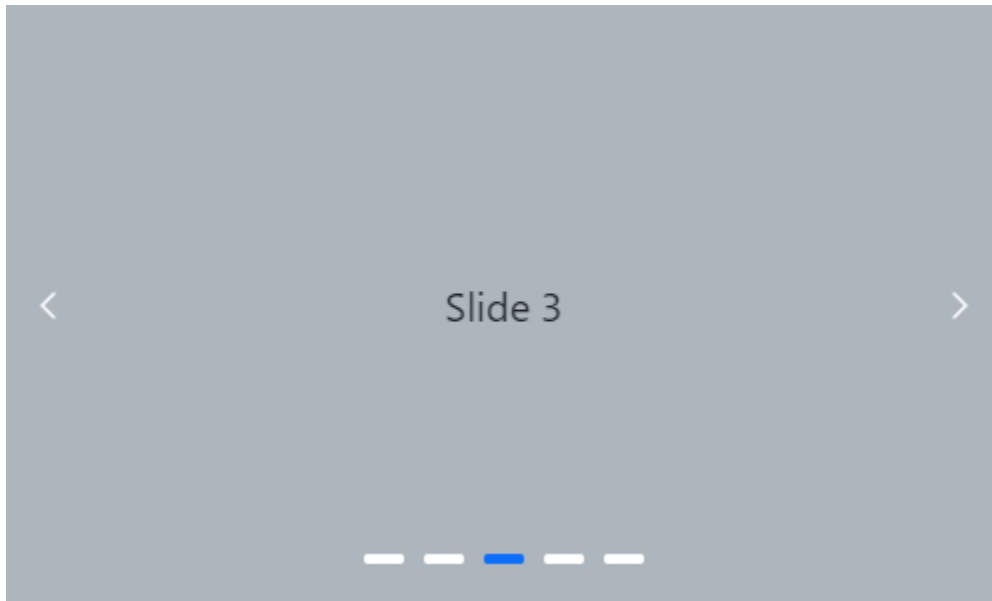
CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel").SelectedIndex(2).Items(new
List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
        })
        .Render()
    )
</div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
```

```
return View();
}
```



Select an item using the method

Using the `prev` or `next` public method of the Carousel component, you can switch the current populating slide to a previous or next slide.

CSHTML

```
@using Syncfusion.EJ2.Buttons;
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Button("prev").Content("Prev").Render())
@(Html.EJS().Button("next").Content("Next").Render())
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel").Items(new
List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
        })
        .Render()
    )
    </div>
</div>
<script>
    document.getElementById('prev').onclick = function () {
        var carouselObj = document.querySelector(".e-
carousel").ej2_instances[0];
```

```

        carouselObj.prev();
    }
    document.getElementById('next').onclick = function () {
        var carouselObj = document.querySelector(".e-
carousel").ej2_instances[0];
        carouselObj.next();
    }
</script>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```

Partial visible slides

The Carousel component supports to show one complete slide and a partial view of adjacent (previous and next) slides at the same time. You can enable or disable the partial slides using the [partialVisible](#) property.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").PartialVisible(true).Items(new
List<CarouselItem> {
    new CarouselItem { Template = "#itemTemplate1" },
    new CarouselItem { Template = "#itemTemplate2" },
    new CarouselItem { Template = "#itemTemplate3" },
    new CarouselItem { Template = "#itemTemplate4" },
    new CarouselItem { Template = "#itemTemplate5" }
}))
        .Render()
    </div>
</div>
<script id="itemTemplate1" type="text/x-template">

```

```

    <figure class="img-container">
        
        <figcaption class="img-caption">Golden Gate Bridge, San
Francisco</figcaption>
    </figure>
</script>
<script id="itemTemplate2" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Spring Flower Trees</figcaption>
    </figure>
</script>
<script id="itemTemplate3" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Oddadalen Waterfalls,
Norway</figcaption>
    </figure>
</script>
<script id="itemTemplate4" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Anse Source d'Argent,
Seychelles</figcaption>
    </figure>
</script>
<script id="itemTemplate5" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Stonehenge, England</figcaption>
    </figure>
</script>
<style>
    .control-container {
        background-color: #e5e5e5;
        height: 360px;
        margin: 0 auto;
        width: 600px;
    }

    .img-container {
        height: 100%;
        margin: 0;
    }

    .img-caption {
        color: #fff;
        font-size: 1rem;
        position: absolute;
        bottom: 3rem;
        width: 100%;
        text-align: center;
    }

```

```
}
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```

Note: Slide animation only applicable if the `partialVisible` is enabled.

The last slide will be displayed as a partial slide at the initial rendering when the `loop` and `partialVisible` properties are enabled.

The previous slide is not displayed at the initial rendering when the `loop` is disabled.

The following example code depicts the functionality of `partialVisible` and without `loop` functionalities.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").PartialVisible(true).Loop(false).Items(
new List<CarouselItem> {
    new CarouselItem { Template = "#itemTemplate1" },
    new CarouselItem { Template = "#itemTemplate2" },
    new CarouselItem { Template = "#itemTemplate3" },
    new CarouselItem { Template = "#itemTemplate4" },
    new CarouselItem { Template = "#itemTemplate5" }
})
    .Render()
)
    </div>
</div>
<script id="itemTemplate1" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Golden Gate Bridge, San
Francisco</figcaption>
    </figure>
</script>
<script id="itemTemplate2" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Spring Flower Trees</figcaption>
    </figure>
</script>
<script id="itemTemplate3" type="text/x-template">
    <figure class="img-container">
        
```

```

        <figcaption class="img-caption">Oddadalen Waterfalls,
Norway</figcaption>
    </figure>
</script>
<script id="itemTemplate4" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Anse Source d'Argent,
Seychelles</figcaption>
    </figure>
</script>
<script id="itemTemplate5" type="text/x-template">
    <figure class="img-container">
        
        <figcaption class="img-caption">Stonehenge, England</figcaption>
    </figure>
</script>
<style>
    .control-container {
        background-color: #e5e5e5;
        height: 360px;
        margin: 0 auto;
        width: 600px;
    }

    .img-container {
        height: 100%;
        margin: 0;
    }

    .img-caption {
        color: #fff;
        font-size: 1rem;
        position: absolute;
        bottom: 3rem;
        width: 100%;
        text-align: center;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```

See also

- [Customizing partial slides area size](#)

Animations and Transitions

Animations

Fade animation

In Carousel, two built-in animations are provided for slide transitions. You can disable animation using the [AnimationEffect](#) property. By default, Slide animation is applied for the transition between slides.

The following demo depicts the example for **Fade** animation,

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="control-container">

@(Html.EJS().Carousel("defaultCarousel").AnimationEffect(CarouselAnimationEf
fect.Fade)
    .Items(new List<CarouselItem> {
        new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
        new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
        new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
        new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
        new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
    })
    .Render()
)
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```



Custom animation

In Carousel, you can use customized animation effects for slide transitions using the [Custom](#) option of the [AnimationEffect](#) property and apply custom animation css via [cssClass](#) property.

The following demo depicts the example for **parallax** custom animation,

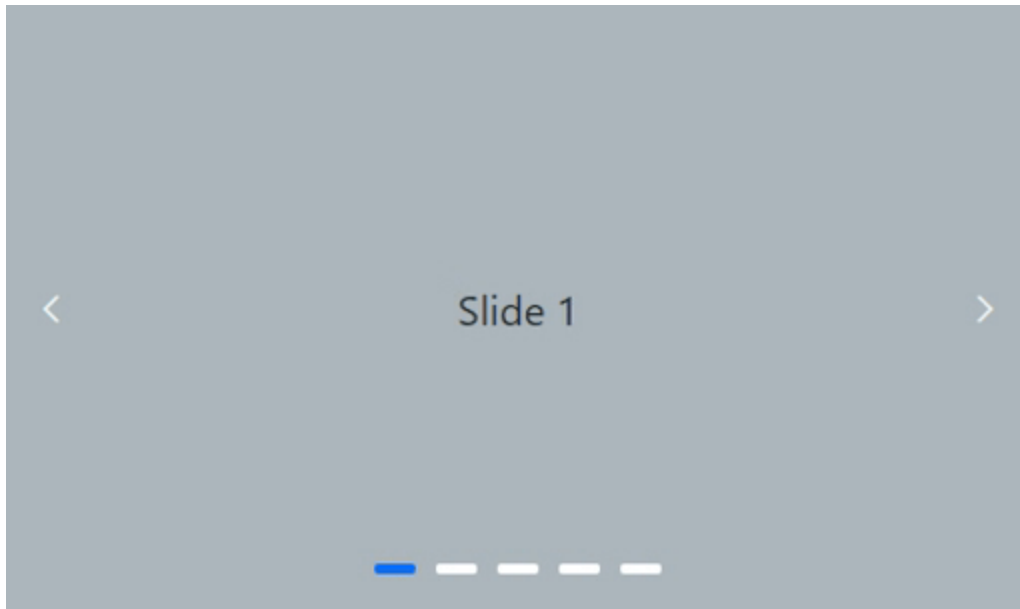
CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="control-container">
    @(Html.EJS().Carousel("defaultCarousel").
        AnimationEffect(CarouselAnimationEffect.Custom).CssClass("parallax")
        .Items(new List<CarouselItem>
        {
            new CarouselItem { Template = "<div class='slide-content'>Slide
1</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
        })
        .Render()
    )
</div>
<style>
    .control-container {
        background-color: #adb5bd;
    }
</style>
```

```
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
    /* Parallax animation */
    .parallax .e-carousel-item {
        transition: transform 1s ease-in-out;
    }
    .parallax .e-carousel-item.e-next {
        animation: ParallaxIn 1s ease-in-out;
    }
    .parallax .e-carousel-item.e-prev {
        animation: ParallaxOut 1s ease-in-out;
    }
    @@keyframes ParallaxIn {
        from {
            opacity: 0;
            transform: scale(0) translateY(100%);
        }
        to {
            opacity: 1;
            transform: scale(1) translateY(0);
        }
    }
    @@keyframes ParallaxOut {
        from {
            opacity: 1;
            transform: scale(1) translateY(0);
        }
        to {
            opacity: 0;
            transform: scale(0) translateY(-100%);
        }
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```



Intervals between slides

Using the items property, you can set different intervals for each item to transition between slides. The default interval is 5000 ms (5 seconds). The following example depicts the code for setting the different intervals between each item.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel").Items(new
List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>", Interval = 3000 },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>", Interval = 2000 },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>", Interval = 1000 },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>", Interval = 5000 },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>", Interval = 6000 }
        })
        .Render()
    )
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
    }
</style>
```

```

        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```

Note: Interval property can accept value in terms of milliseconds.

Auto play slides

In the carousel, all slides transitions are performed continuously after the specified or default intervals. You can enable or disable the auto slide transition using the [AutoPlay](#) property. The following example depicts the code to disable the auto slide transitions.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel").AutoPlay(false).Items(new
List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-content'>Slide
1</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
            new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
        })
        .Render()
    )
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
    }

```

```
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```

Pause on hover

By default, Slide transitions are paused when hovering the mouse pointer over the Carousel element. You can enable or disable this functionality using the [PauseOnHover](#) property.

The following example depicts the code to play the slides when hovering the mouse pointer over the Carousel element.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").PauseOnHover(false).Items(new
List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
}))
    .Render()
    )
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
```

```
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```

Looping slides

In the carousel, slides transitions are repeated continuously when you reach the last slide by default. You can enable or disable the infinite slide transition using the [Loop](#) property. The following example depicts the code to disable the infinite slide transitions.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel").Loop(false).Items(new
List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
        })
        .Render()
    )
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
```

```
{
    return View();
}
```



Slide changing events

Using the [SlideChanging](#) or [SlideChanged](#) events of the Carousel component, you can perform sample end customization while the carousel items are switched.

The following demo depicts the example for carousel events,

CSSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel")
            .SlideChanging("onSlideChanging")
            .SlideChanged("onSlideChanged")
            .Items(new List<CarouselItem> {
                new CarouselItem { Template = "<div class='slide-content'>Slide 1</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 2</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 3</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 4</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 5</div>" }
            })
            .Render()
        )
    </div>
</div>
<script type="text/javascript">
    function onSlideChanging(args) {
```



```
        console.log(args.currentSlide); // You can customize the slide
before changing
    }
    function onSlideChanged(args) {
        console.log(args.currentSlide); // You can customize the slide after
changed
    }
</script>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class AccordionController : Controller
    {
        // GET: //
        public IActionResult DefaultFunctionalities()
        {
            return View();
        }
        public ActionResult PartialView1()
        {
            return PartialView();
        }
        public ActionResult PartialView2()
        {
            return PartialView();
        }
    }
}
```

Disable touch swiping

In the carousel, you can swipe the carousel slides using touch actions by default. The swipe action can be enabled or disabled using the [EnableTouchSwipe](#) property. The following example depicts the code to disable the swipe action for the slide.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").EnableTouchSwipe(false).Items(new
List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
})
    .Render()
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```

Swipe Modes

In the carousel, the [swipeMode](#) property allows specifying whether the slide transition should occur while performing swiping via touch or mouse. The slide swiping is enabled or disabled using the bitwise operator.

The following are the different swipe modes available in the carousel:

- CarouselSwipeMode.Touch - Allows the user to slide the slides using touch actions.
- CarouselSwipeMode.Mouse - Allows the user to slide the slides using mouse actions.
- CarouselSwipeMode.Touch & CarouselSwipeMode.Mouse - Allows the user to slide the slides using both touch and mouse actions.
- ~CarouselSwipeMode.Touch & ~CarouselSwipeMode.Mouse - Disables both touch and mouse actions.

CSHTML

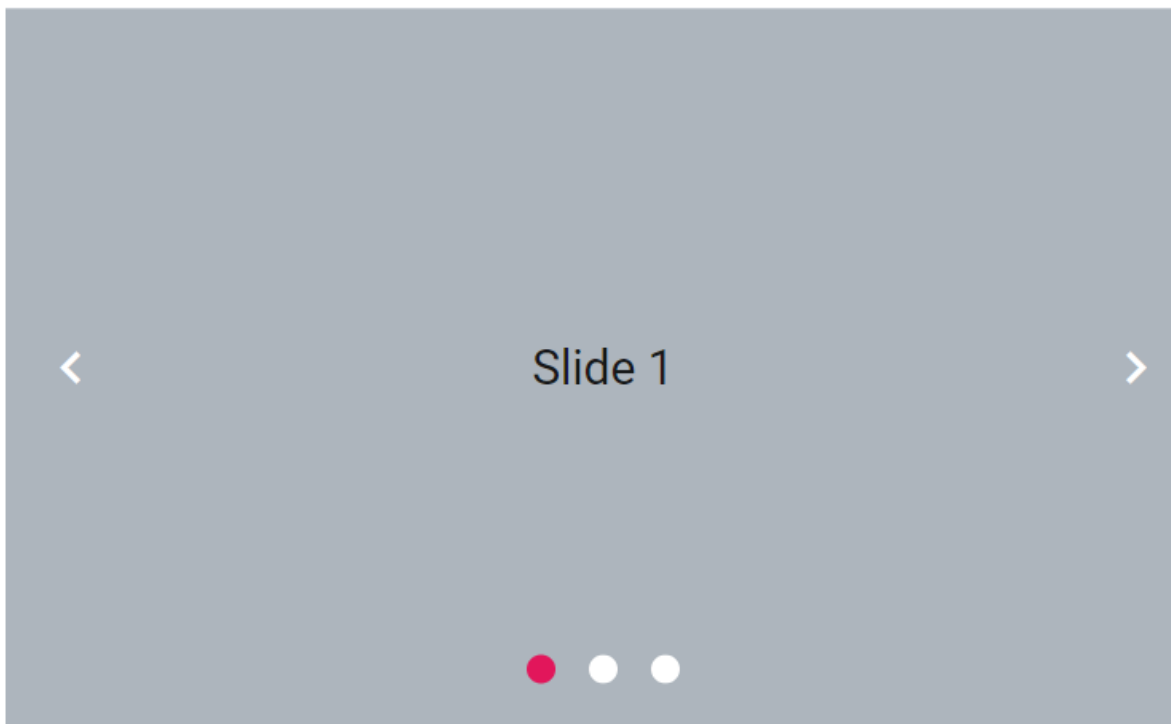
```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").AutoPlay(false).SwipeMode(CarouselS
wipeMode.Mouse & CarouselSwipeMode.Touch).Items(new List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-content'>Slide
1</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
    })
    .Render()
)

    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()  
{  
    return View();  
}
```



Navigators and Indicators

The navigators and indicators are used to transition the slides manually.

Navigators

Show or hide previous and next button

In navigators, the previous and next slide transition buttons are used to perform slide transitions manually. You can show/hide the navigators using the [ButtonsVisibility](#) property. The possible property values are as follows:

- **Hidden** – the navigator's buttons are not visible.
- **Visible** – the navigator's buttons are visible.
- **VisibleOnHover** – the navigator's buttons are visible only when hovering over the carousel.

The following example depicts the code to hide the navigators in the carousel.

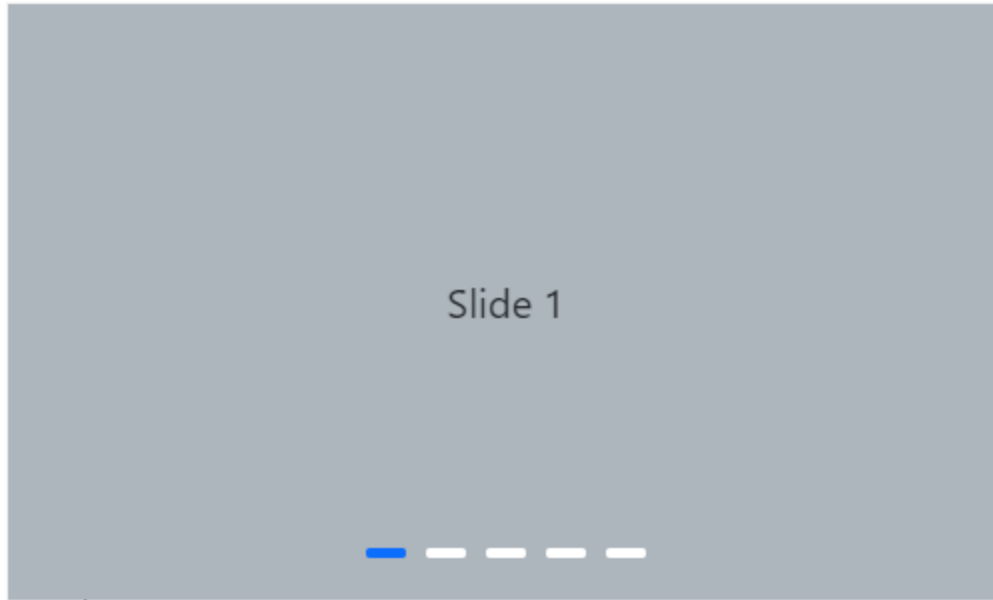
CSHTML

```
@using Syncfusion.EJ2.Navigations;  
<div class="container">  
    <div class="control-container">  
  
@ (Html.EJS().Carousel("defaultCarousel").ButtonsVisibility(CarouselButtonVis  
ibility.Hidden).Items(new List<CarouselItem> {
```

```
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 1</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 2</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 3</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 4</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 5</div>" }  
    })  
    .Render()  
)  
</div>  
</div>  
<style>  
    .control-container {  
        background-color: #adb5bd;  
        height: 300px;  
        margin: 0 auto;  
        width: 500px;  
    }  
    .e-carousel .slide-content {  
        align-items: center;  
        display: flex;  
        font-size: 1.25rem;  
        height: 100%;  
        justify-content: center;  
    }  
</style>
```

DATA.CS

```
public ActionResult Index()  
{  
    return View();  
}
```



Show previous and next button on hover

In the carousel, you can show the previous and next buttons only on mouse hover using the [ButtonsVisibility](#) property. The following example depicts the code to show the navigators on mouse hover in the carousel.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").ButtonsVisibility(CarouselButtonVis-
ibility.VisibleOnHover).Items(new List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
}))
    .Render()
    )
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
```

```

        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

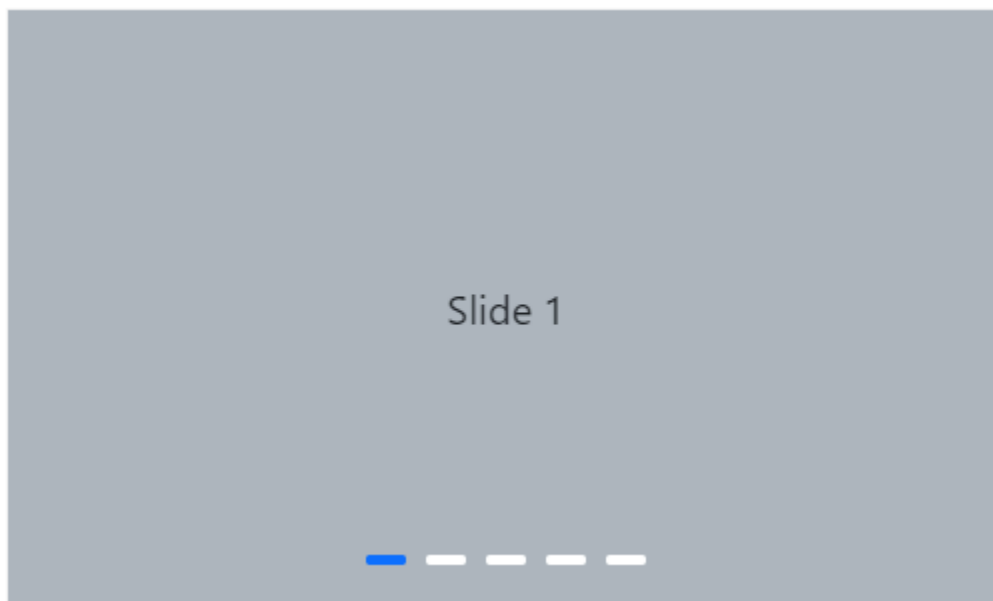
```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```



Previous and next button template

Template options are provided to customize the previous button using [PreviousButtonTemplate](#) and the next button using [NextButtonTemplate](#). The following example depicts the code for applying the template to previous and next buttons in the carousel.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel")
            .PreviousButtonTemplate("<button id='previous'></button>")
            .NextButtonTemplate("<button id='next'></button>")
            .Items(new List<CarouselItem> {
                new CarouselItem { Template = "<div class='slide-content'>Slide 1</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 2</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 3</div>" },
            })
        )
    </div>
</div>

```

```

        new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
        new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
    })
    .Render()
)
</div>
</div>
<script type="text/javascript">
    document.addEventListener('DOMContentLoaded', function () {
        var prevBtn = new ej.buttons.Button({ cssClass: 'e-flat e-round',
iconCss: 'e-icons e-chevron-left-double' });
        prevBtn.appendTo('#previous');
        var nextBtn = new ej.buttons.Button({ cssClass: 'e-flat e-round',
iconCss: 'e-icons e-chevron-right-double' });
        nextBtn.appendTo('#next');
    });
</script>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class AccordionController : Controller
    {
        // GET: //
        public IActionResult DefaultFunctionalities()
        {
            return View();
        }
        public ActionResult PartialView1()
        {
            return PartialView();
        }
    }
}

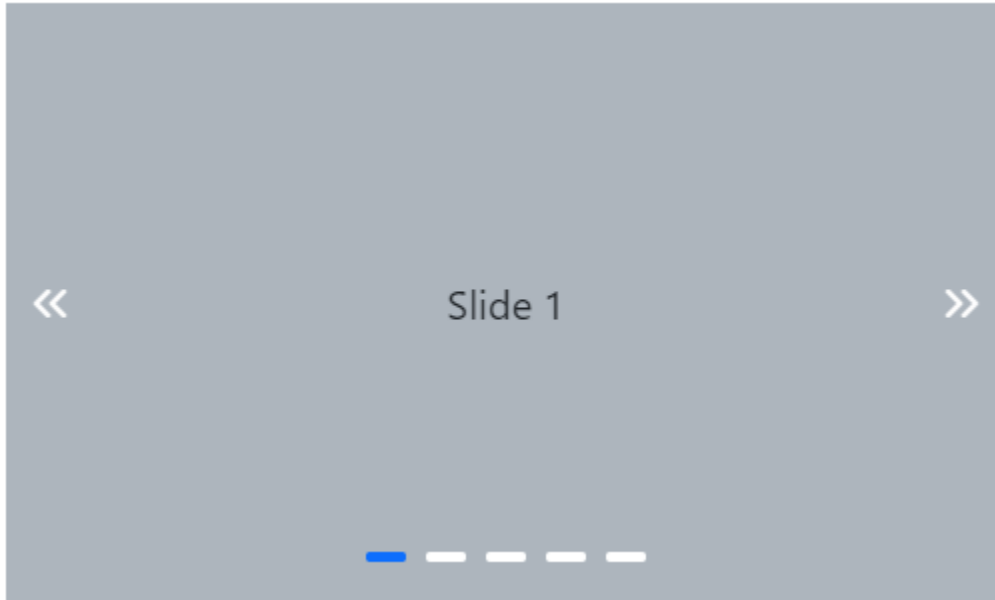
```



```

public ActionResult PartialView2()
{
    return PartialView();
}
}

```



Indicators

Show or hide indicators

In indicators, the total slides and current slide state have been depicted. You can show/hide the indicators using the [ShowIndicators](#) property. The following example depicts the code to hide the indicators in the carousel.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").ShowIndicators(false).Items(new
List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
    new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
}))
    .Render()
    )
</div>

```

```

</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```



Indicators template

Template option is provided to customize the indicators by using the [IndicatorsTemplate](#) property. The following example depicts the code for applying a template to indicators in the carousel.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel")

```

```

        .IndicatorsTemplate("<div class='indicator' indicator-
index='${index}'></div>")
        .Items(new List<CarouselItem> {
            new CarouselItem { Template = "<div class='slide-
content'>Slide 1</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 2</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 3</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 4</div>" },
            new CarouselItem { Template = "<div class='slide-
content'>Slide 5</div>" }
        })
        .Render()
    )
</div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
    .e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar.e-
template .indicator {
        background-color: #ececec;
        border-radius: 0.25rem;
        cursor: pointer;
        height: 0.5rem;
        margin: 0.5rem;
        width: 1.5rem;
    }
    .e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar.e-
active .indicator {
        background-color: #3c78ef;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```



Showing preview of slide in indicator

You can customize the indicators by showing the preview image of each slide using the [IndicatorsTemplate](#) property. The following example depicts the code for showing the preview image using a template for indicators in the carousel.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">
        @(Html.EJS().Carousel("defaultCarousel")
            .SlideChanged("onSlideChanged")
            .IndicatorsTemplate("#indicatorTemplate")
            .Items(new List<CarouselItem> {
                new CarouselItem { Template = "<div class='slide-content'>Slide 1</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 2</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 3</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 4</div>" },
                new CarouselItem { Template = "<div class='slide-content'>Slide 5</div>" }
            })
        .Render()
    )
    </div>
</div>
<script type="text/x-template" id="indicatorTemplate">
    <div class="indicator" indicator-index="{index}">
        <div class="preview-content">${getContent(data.index)}</div>
    </div>
</script>
<script type="text/javascript">
    window.getContent = function (index) {
```

```

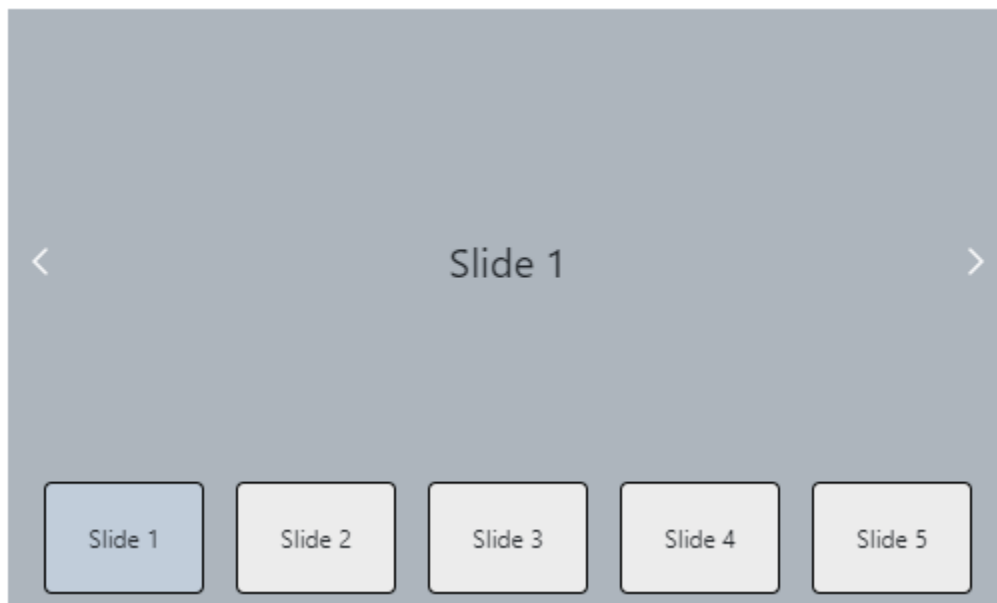
        var birds = ['Slide 1', 'Slide 2', 'Slide 3', 'Slide 4', 'Slide 5'];
        return birds[index];
    };
    function onSlideChanged(args) {
        var carouselObj = document.querySelector(".e-
carousel").ej2_instances[0];
        var indicators = carouselObj.element.querySelector('.e-carousel-
indicators');
        ej.base.removeClass(indicators.querySelectorAll('.indicator'),
'active');
        ej.base.addClass([(indicators.querySelector('[data-index="' +
args.currentIndex + '"').children[0])], 'active');
    }
</script>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
    .e-carousel .e-carousel-items,
    .e-carousel .e-carousel-navigators {
        height: calc(100% - 3rem);
    }
    .e-carousel .e-carousel-navigators .e-previous,
    .e-carousel .e-carousel-navigators .e-next,
    .e-carousel .e-carousel-navigators .nav-btn {
        padding: 0;
    }
    .e-carousel .e-carousel-navigators .nav-btn:active,
    .e-carousel .e-carousel-navigators .nav-btn:focus,
    .e-carousel .e-carousel-navigators .nav-btn:hover {
        background-color: transparent !important;
        color: inherit;
    }
    .e-carousel .e-carousel-navigators svg {
        fill: none;
        stroke: currentColor;
        stroke-linecap: square;
        stroke-width: 8px;
        height: 2rem;
        vertical-align: middle;
        width: 2rem;
    }
    .e-carousel .e-carousel-navigators .e-previous svg {
        transform: rotate(180deg);
    }
    .e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar
.indicator {

```

```
background-color: #ecec;
border: 1px solid black;
border-radius: 0.25rem;
cursor: pointer;
height: 3.5rem;
margin: 0.5rem;
width: 5rem;
}
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar.e-
active .indicator {
background-color: #c1cdda;
}
.preview-content {
align-items: center;
display: flex;
height: 100%;
justify-content: center;
}
</style>
```

DATA.CS

```
public ActionResult Index()
{
return View();
}
```



Indicators Types

Choose different types of indicators available using the [IndicatorsType](#) property. The indicator types are categorized as follows:

- [Default Indicator](#)
- [Dynamic Indicator](#)
- [Fraction Indicator](#)

- [Progress Indicator](#)

Default Indicator

A default indicator in a carousel is a set of dots that indicate the current position of the slide in the carousel. The Default indicator can be achieved by setting the [IndicatorsType](#) to **Default**.

CSHTML

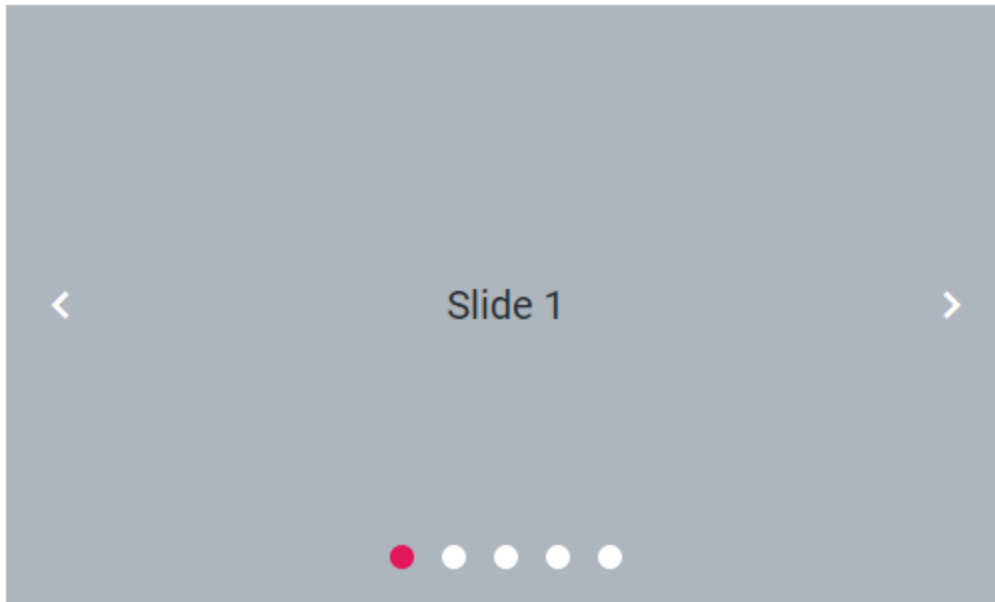
```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").AutoPlay(false).IndicatorsType(CarouselIndicatorsType.Default).Items(new List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-content'>Slide
1</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
}))
    .Render()
)

    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```



Dynamic Indicator

A dynamic indicator in a carousel provides visual cues or markers that dynamically change or update to indicate the current position. The Dynamic indicator can be achieved by setting the [IndicatorsType](#) to **Dynamic**.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").AutoPlay(false).IndicatorsType(CarouselIndicatorsType.Dynamic).Items(new List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-content'>Slide
1</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
}))
    .Render()
)
    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
    }
</style>
```



```

        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

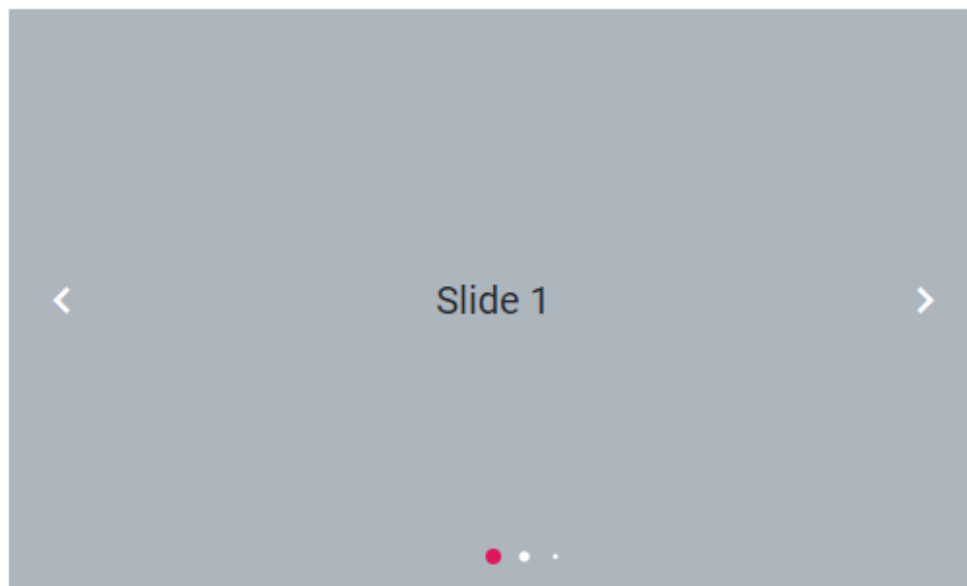
```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```



Fraction Indicator

The fraction indicator type displays the current slide index and total slide count as a fraction. The Fraction indicator can be achieved by setting the [IndicatorsType](#) to Fraction.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").AutoPlay(false).IndicatorsType(CarouselIndicatorsType.Fraction).Items(new List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-content'>Slide 1</div>" },

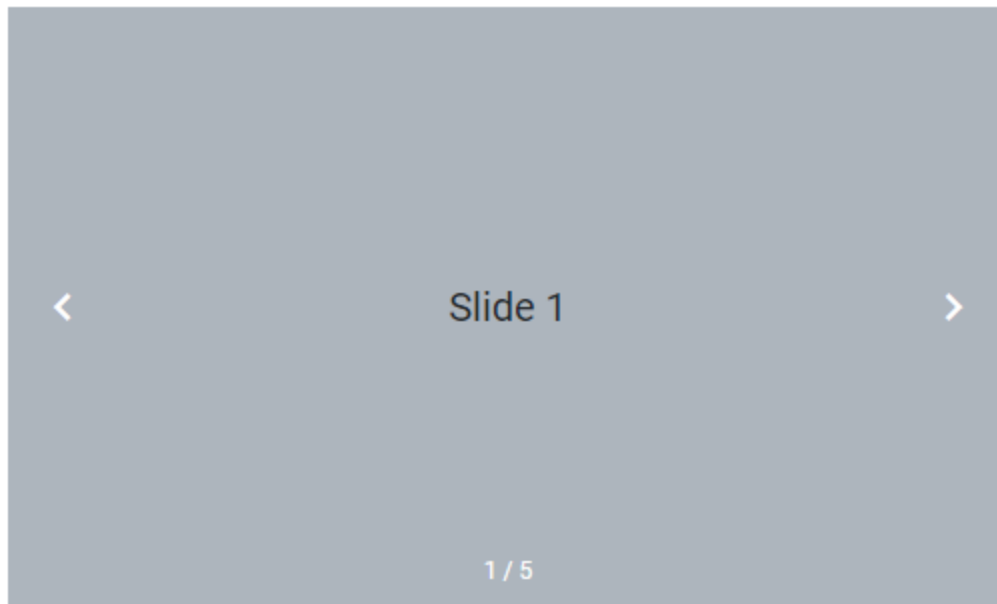
```

```
        new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
        new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
        new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
        new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
    })
    .Render()
)

</div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>
```

DATA.CS

```
public ActionResult Index()
{
    return View();
}
```



Progress Indicator

The Progress Indicator type displays the current slide as a progress bar. The Progress indicator can be achieved by setting the [IndicatorsType](#) to `Progress`.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").AutoPlay(false).IndicatorsType(CarouselIndicatorsType.Progress).Items(new List<CarouselItem> {
    new CarouselItem { Template = "<div class='slide-content'>Slide
1</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
2</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
3</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
4</div>" },
    new CarouselItem { Template = "<div class='slide-content'>Slide
5</div>" }
    })
    .Render()
)

    </div>
</div>
<style>
    .control-container {
        background-color: #adb5bd;
        height: 300px;
        margin: 0 auto;
    }
</style>
```

```

        width: 500px;
    }
    .e-carousel .slide-content {
        align-items: center;
        display: flex;
        font-size: 1.25rem;
        height: 100%;
        justify-content: center;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```



Play button

Show or hide the play button

In the carousel, [AutoPlay](#) actions have been controlled by using the [ShowPlayButton](#) property in the user interface. When you enable this property, the slide transitions are controlled using this play and pause button. This property depends on the [ButtonsVisibility](#) property. The following example depicts the code to show the play button in the carousel.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="container">
    <div class="control-container">

@ (Html.EJS().Carousel("defaultCarousel").ShowPlayButton(true).Items(new
List<CarouselItem> {

```

```
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 1</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 2</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 3</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 4</div>" },  
        new CarouselItem { Template = "<div class='slide-  
content'>Slide 5</div>" }  
    })  
    .Render()  
)  
</div>  
</div>  
<style>  
    .control-container {  
        background-color: #adb5bd;  
        height: 300px;  
        margin: 0 auto;  
        width: 500px;  
    }  
    .e-carousel .slide-content {  
        align-items: center;  
        display: flex;  
        font-size: 1.25rem;  
        height: 100%;  
        justify-content: center;  
    }  
</style>
```

DATA.CS

```
public ActionResult Index()  
{  
    return View();  
}
```



Play button template

Template option is provided to customize the play button by using the [PlayButtonTemplate](#) property. The following example depicts the code for applying a template to play Button in the carousel.

CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="control-container">
    @(Html.EJS().Carousel("defaultCarousel")
        .ShowPlayButton(true)
        .PlayButtonTemplate("<div id='play'></div>")
        .Items(new List<CarouselItem> {
            new CarouselItem { Template = "<figure class='img-
container'><img
src='https://ej2.syncfusion.com/products/images/carousel/cardinal.png'
alt='cardinal' style='height:100%;width:100%;' /><figcaption class='img-
caption'>Cardinal</figcaption></figure>" },
            new CarouselItem { Template = "<figure class='img-
container'><img
src='https://ej2.syncfusion.com/products/images/carousel/hunei.png'
alt='kingfisher' style='height:100%;width:100%;' /><figcaption class='img-
caption'>Kingfisher</figcaption></figure>" },
            new CarouselItem { Template = "<figure class='img-
container'><img
src='https://ej2.syncfusion.com/products/images/carousel/costa-rica.png'
alt='keel-billed-toucan' style='height:100%;width:100%;' /><figcaption
class='img-caption'>Keel-billed-toucan</figcaption></figure>" },
            new CarouselItem { Template = "<figure class='img-
container'><img
src='https://ej2.syncfusion.com/products/images/carousel/kaohsiung.png'
alt='yellow-warbler' style='height:100%;width:100%;' /><figcaption
class='img-caption'>Yellow-warbler</figcaption></figure>" },
            new CarouselItem { Template = "<figure class='img-
container'><img
src='https://ej2.syncfusion.com/products/images/carousel/bee-eater.png'
alt='bee-eater' style='height:100%;width:100%;' /><figcaption class='img-
caption'>Bee-eater</figcaption></figure>" }
        })
    )
</div>
```

```

alt='bee-eater' style='height:100%;width:100%;' /><figcaption class='img-
caption'>Bee-eater</figcaption></figure>" }
    })
    .Render()
)
</div>
<script type="text/javascript">
    document.addEventListener('DOMContentLoaded', function () {
        var carouselObj = document.querySelector(".e-
carousel").ej2_instances[0];
        var button = new ej.buttons.Button({ cssClass: 'e-play-pause-btn',
content: "Pause" });
        button.appendTo('#play');
        button.element.onclick = () => {
            if (carouselObj.autoPlay) {
                button.content = "Play";
                carouselObj.autoPlay = false;
            } else {
                button.content = "Pause";
                carouselObj.autoPlay = true;
            }
        };
    });
</script>
<style>
    .control-container {
        height: 360px;
        margin: 0 auto;
        width: 600px;
    }
    .img-container {
        height: 100%;
        margin: 0;
    }
    .img-caption {
        color: #fff;
        font-size: 1rem;
        position: absolute;
        bottom: 3rem;
        width: 100%;
        text-align: center;
    }
    .e-play-pause-btn {
        border-radius: unset !important;
        color: white !important;
    }
</style>

```

DATA.CS

```

public ActionResult Index()
{
    return View();
}

```



Accessibility in Carousel control

The Carousel control has been designed, keeping in mind the [WAI-ARIA](#) specifications, and applying the WAI-ARIA roles, states and properties along with keyboard support for people who use assistive devices. WAI-ARIA accessibility support is achieved through attributes like `aria-roledescription`, `aria-label`, `aria-current`, `aria-live`, `aria-role` and `aria-hidden`. It provides information about elements in a document for assistive technology. The control implements keyboard navigation support by following the [WAI-ARIA practices](#) and has been tested in major screen readers.

The accessibility compliance for the Carousel control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | `` |

| [Section 508](#) Support | `` |

| Screen Reader Support | `` |

| Right-To-Left Support | `` |

| Color Contrast | `` |

| Mobile Device Support | `` |


```

| Keyboard Navigation Support |  |

| Accessibility Checker Validation |  |

| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

```

ARIA attributes

The Carousel control is designed by considering [WAI-ARIA](#) standard. Carousel is supported with ARIA Accessibility which is accessible by on-screen readers and other assistive technology devices. The following list of attributes is added to the Carousel.

| Roles and Attributes | Functionalities

aria-roledescription	The role description attribute has been added for the root element (carousel) and each carousel slide item (slide).
aria-label	Previous, next and play/pause buttons and all indicator elements.
aria-current	For the active item indicator element, <code>aria-current</code> is set to <code>true</code> .
aria-hidden	For all carousel elements except the currently visible item, <code>aria-hidden</code> is set to <code>true</code> .
aria-live	For carousel items element, when <code>autoPlay</code> is <code>true</code> , <code>aria-live</code> is set to <code>off</code> ; when <code>autoPlay</code> is <code>false</code> , <code>aria-live</code> is set to <code>polite</code> .
aria-role	For carousel slide item, <code>aria-role</code> has been grouped.

Keyboard interaction

By default, keyboard navigation is enabled. This control implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Carousel element, you can use the following key combination for interacting with the Carousel.

Key	Description
Alt + J	Keys to focus the Carousel control (done at application end).
Arrows	Keys to navigate between slides.
Home	To navigate to the first slide.
End	To navigate to the last slide.
Space	To play/pause the slide transitions.
Enter	To perform the respective action on its focus.
Tab	To Move focus through the interactive elements.
Shift + Tab	To Move focus through the interactive elements.

Ensuring accessibility

The Carousel control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Carousel control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Carousel control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Styles and Appearances in ASP.NET MVC Carousel Component

To modify the Carousel appearance, you need to override the default CSS of Carousel component. Find the list of CSS classes and its corresponding section in Carousel component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Structure in ASP.NET MVC Carousel Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Css class	Purpose
.e-carousel .e-carousel-item	To customize the carousel item
.e-carousel-item.e-active	To customize the active carousel item
.e-carousel .e-carousel-indicators	To customize the indicators
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar	To customize the indicator bars

|.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar .e-indicator|To customize the individual indicator appearance

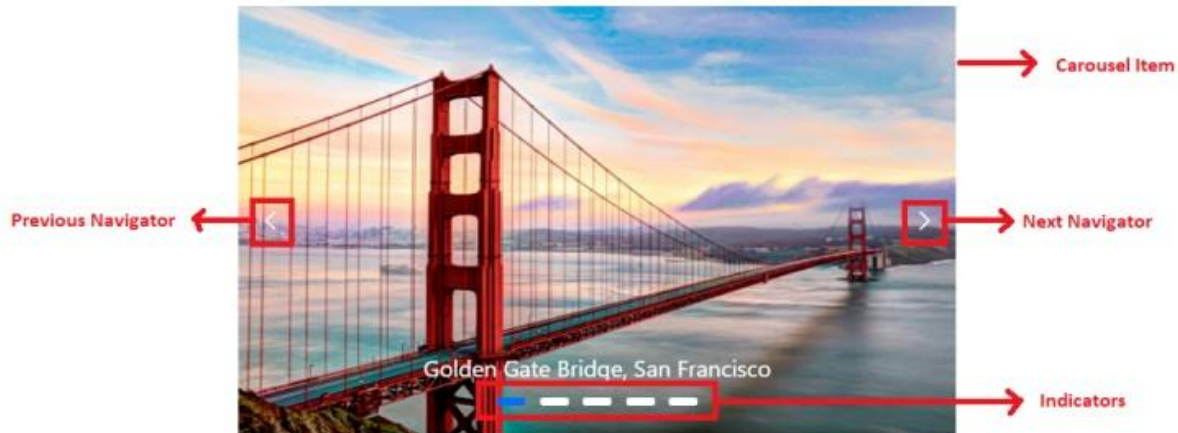
|.e-carousel .e-carousel-navigators|To customize the navigators

|.e-carousel .e-carousel-navigators .e-previous|To customize the previous button

|.e-carousel .e-carousel-navigators .e-next|To customize the next button

|.e-carousel .e-carousel-navigators .e-play-pause|To customize the play and pause button

|.e-carousel.e-partial .e-carousel-slide-container|To customize the partial visible slides



Customizing the indicators

Use the following CSS to customize the space between indicators by overriding the `.e-indicator-bar` CSS class.

`CSS

```
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar {
padding: 8px;
}
`
```



Use the following CSS to customize the indicators appearance by overriding the `.e-indicator` CSS class.

```
`CSS
```

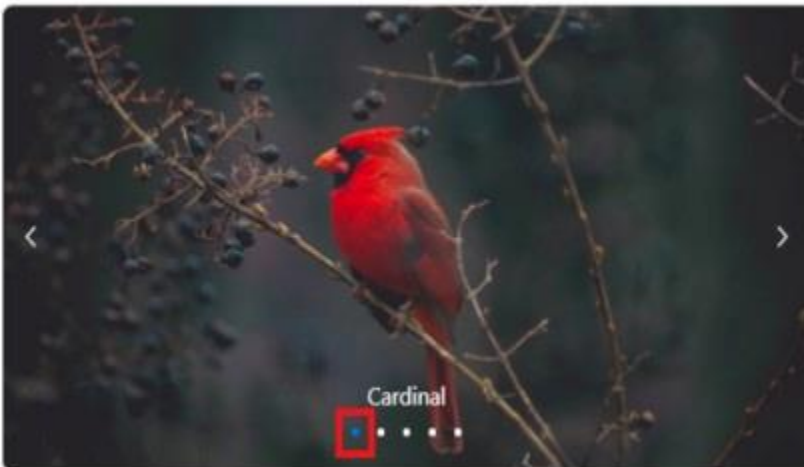
```
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar .e-indicator {
```

```
width: 20px;
```

```
border-radius: 100%;
```

```
}
```

```
,
```



Use the following CSS to render the indicators outside the carousel items by overriding the `.e-carousel-indicators` CSS class.

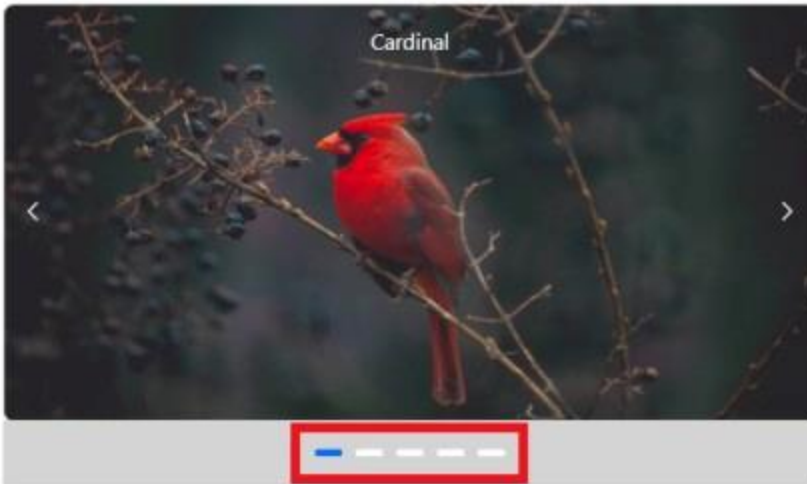
```
`CSS
```

```
.e-carousel .e-carousel-indicators {
```

```
bottom: auto;
```

```
}
```

```
,
```



Customizing the navigators

Use the following CSS to customize the previous and next icon size and colors.

`CSS

```
.e-carousel .e-carousel-navigators .e-next .e-btn:not(:disabled) .e-btn-icon,  
.e-carousel .e-carousel-navigators .e-previous .e-btn:not(:disabled) .e-btn-icon  
{  
  color: greenyellow;  
  font-size: 25px;  
}
```



Use the following CSS to customize the navigators position to bottom by overriding the `.e-carousel-navigators` CSS class.

`CSS

```
.e-carousel .e-carousel-navigators {
```

```
top: 120px;
```

```
}
```

```
,
```



Use the following CSS to render the previous and next icon to outside the carousel items by overriding the `.e-previous` and `.e-next` CSS class.

```
`CSS
```

```
.e-carousel .e-carousel-navigators .e-previous,
```

```
.e-carousel .e-carousel-navigators .e-next
```

```
{
```

```
margin: -60px;
```

```
background: black;
```

```
}
```

```
,
```



Customizing partial slides size

You can customise the partial slide size by overriding the `.e-carousel-slide-container` CSS class.

`CSS

```
.e-carousel.e-partial .e-carousel-slide-container{
padding: 0 150px;
}
`
```



Chart

Getting Started with ASP.NET MVC Chart Control

This section briefly explains about how to include [ASP.NET MVC Chart](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5

NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

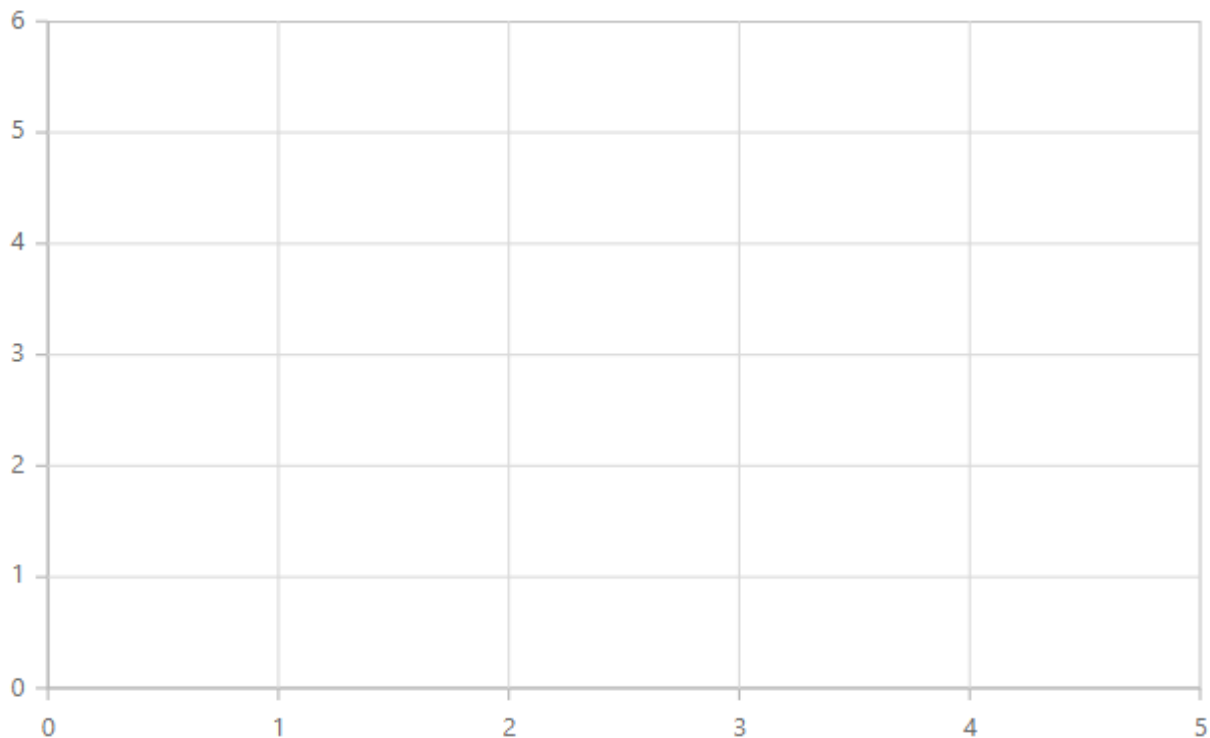
Add ASP.NET MVC Chart control

Now, add the Syncfusion ASP.NET MVC Chart control in **~/Home/Index.cshtml** page.

CSHTML

```
@Html.EJS().Chart("container").Render()
```


Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Chart control will be rendered in the default web browser.



Note: [View Sample in GitHub.](#)

<!-- markdownlint-disable MD036 -->

Working with Data

Chart can visualise data bound from local or remote data.

Local Data

You can bind a simple JSON data to the chart using [DataSource](#) property in series. Now map the fields in JSON to [XName](#) and [YName](#) properties.

CSHTML

```
@Html.EJS().Chart("container").
ChartArea(area => area.Border(ViewBag.chartBorder)).
Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("x").
YName("y").DataSource("series1").Name("Product X").Add();
}).
PrimaryXAxis(px =>
px.Skeleton("y").Title("Years").MajorGridLines(ViewBag.majorGridLines).
ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).
LabelFormat("y").
PrimaryYAxis(py => py.Title("Price").
```

```

LineStyle(ViewBag.lineStyle).
MajorTickLines(ViewBag.majorTickLines).
MinorTickLines(ViewBag.minorTickLines).
RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).

LabelFormat("${value}").EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPl
acement.Shift).
Title("Profit").
Title("Stock Price Analysis").
LegendSettings(lg => lg.Visible(true)).Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > .5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1960, (i + 1), i), y: Math.round(value) };
        series1.push(point1);
    }
</script>

```

LOCAL-DATA.CS

```

public IActionResult Index()
{
    List<LineData> chartData = new List<LineData>
    {
        new LineData { xValue = new DateTime(2005, 01, 01), yValue =
21, yValue1 = 28 },
        new LineData { xValue = new DateTime(2006, 01, 01), yValue =
24, yValue1 = 44 },
        new LineData { xValue = new DateTime(2007, 01, 01), yValue =
36, yValue1 = 48 },
        new LineData { xValue = new DateTime(2008, 01, 01), yValue =
38, yValue1 = 50 },
        new LineData { xValue = new DateTime(2009, 01, 01), yValue =
54, yValue1 = 66 },
        new LineData { xValue = new DateTime(2010, 01, 01), yValue =
57, yValue1 = 78 },
        new LineData { xValue = new DateTime(2011, 01, 01), yValue =
70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class LineData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

```
}
```

Common Datasource

You can also bind a JSON data common to all series using [DataSource](#) property in chart.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).Add();
})).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Render()
)
```

COMMON-DATA.CS

```
public ActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
        new GroupingChartData { month= "Jan", sales= 35, sales1= 28 },
        new GroupingChartData { month= "Feb", sales= 28, sales1= 35 },
        new GroupingChartData { month= "Mar", sales= 34, sales1= 32 },
        new GroupingChartData { month= "Apr", sales= 32, sales1= 34 },
        new GroupingChartData { month= "May", sales= 40, sales1= 32 },
        new GroupingChartData { month= "Jun", sales= 32, sales1= 40 },
        new GroupingChartData { month= "Jul", sales= 35, sales1= 55 },
        new GroupingChartData { month= "Aug", sales= 55, sales1= 35 },
        new GroupingChartData { month= "Sep", sales= 38, sales1= 30 },
        new GroupingChartData { month= "Oct", sales= 30, sales1= 38 },
        new GroupingChartData { month= "Nov", sales= 25, sales1= 32 },
        new GroupingChartData { month= "Dec", sales= 32, sales1= 25 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
```

```

    }
    public class GroupingChartData
    {
        public string month;
        public double sales;
        public double sales1;
    }

```

Remote Data

You can also bind remote data to the chart using **DataManager**. The DataManager requires minimal information like webservice URL, adaptor and crossDomain to interact with service endpoint properly. Assign the instance of DataManager to the **DataSource** property in series and map the fields of data to **XName** and **YName** properties. You can also use the **Query** property of the series to filter the data.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("Assignee").Marker(ViewBag.marker).
    YName("Estimate").DataSource("dataManager").
    Query("query").
    Name("Story Point").Add();
}).PrimaryXAxis(px => px.Title("Assignee").
RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).
MajorGridLines(ViewBag.majorGridLines).
ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
ChartArea(area => area.Border(ViewBag.ChartBorder)).
PrimaryYAxis(py => py.MajorGridLines(ViewBag.majorGridLines).
MajorTickLines(ViewBag.majorTickLines).
LineStyle(ViewBag.lineStyle).
Title("Sprint Task Analysis").
LegendSettings(lg => lg.Visible(false)).Render()
}
<script>
    var dataManager = new ej.data.DataManager({
        url: 'https://mvc.syncfusion.com/Services/Northwnd.svc/Tasks/'
    });
    var query = new ej.data.Query().take(5).where('Estimate',
'lessThan', 3, false);
</script>

```

REMOTE-DATA.CS

```

public ActionResult Index()
{
    return View();
}

```

Empty points

The Data points that uses the **null** or **undefined** as value are considered as empty points. Empty data points are ignored and not plotted in the Chart. When the data is provided by using the points property,

By using `EmptyPointSettings` property in series, you can customize the empty point. Default `Mode` of the empty point is `Gap`.

CSHTML

```
@(Html.EJS().Chart("container").ChartArea(ca =>
ca.Border(ViewBag.border))
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
        .Name("Profit")
        .XName("xValue")
        .YName("yValue")
        .Marker(ViewBag.marker)
        .Width(2)
        .EmptyPointSettings(ViewBag.emptyPointSettings)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.Title("Product")
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .Interval(1)
    )
    .Title("Annual Product-Wise Profit Analysis")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").Render()
)
```

EMPTY-POINTS.CS

```
public IActionResult Index()
{
    List<EmptyData> chartData = new List<EmptyData>
    {
        new EmptyData{ month="Jan", sales= 35 },
        new EmptyData{ month="Feb", sales= 28 },
        new EmptyData{ month="Mar", sales=null },
        new EmptyData{ month="Apr", sales=32 },
        new EmptyData{ month="May", sales=40 },
        new EmptyData{ month="Jun", sales=32 },
        new EmptyData{ month="Jul", sales=35 },
        new EmptyData{ month="Aug", sales=null },
        new EmptyData{ month="Sep", sales=38 },
        new EmptyData{ month="Oct", sales=30 },
        new EmptyData{ month="Nov", sales=25 },
        new EmptyData{ month="Dec", sales=32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class EmptyData
{
    public string month;
    public Nullable<double> sales;
}
```

Lazy loading

Lazy loading allows you to load data for chart on demand. Chart will fire the scrollEnd event, in that we can get the minimum and maximum range of the axis, based on this, we can upload the data to chart.

CSHTML

```
@(Html.EJS().Chart("container").Width("100%").Load("load").Title("Network
Load")
    .ScrollEnd("scrollEnd")
    .LegendSettings(le => le.Visible(false))
    .PrimaryXAxis(px => px.ScrollbarSettings(scroll =>
scroll.Enable(true).PointsLength(1000)).Title("Day").ValueType(Syncfusion.EJ
2.Charts.ValueType.DateTime).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLa
belPlacement.Shift).Skeleton("yMMM").SkeletonType(Syncfusion.EJ2.Charts.Skel
etonType.Date))
    .PrimaryYAxis(py => py.Title("Server
Load").LabelFormat("{value}MB")).
    Series(sr => sr.DataSource("GetDateTimeData(new Date(2009, 0,
1), new Date(2009, 8, 1))")
    .XName("x").YName("y").Name("Server
Load").Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Animation(an =>
an.Enable(false)).Add()).
    Tooltip(tl => tl.Enable(true).Shared(true)).Render()
)

<script>
var intl = new ej.s.base.Internationalization();
var chart;
var load = function (args) {
    chart = args.chart;
    args.chart.primaryXAxis.scrollbarSettings.range = {
        minimum: new Date(2009, 0, 1),
        maximum: new Date(2014, 0, 1)
    };
}
function GetDateTimeData(start, end) {
    var series1 = [];
    var date;
    var value = 30;
    var option = {
        skeleton: 'full',
        type: 'dateTime'
    };
    var dateParser = intl.getDateParser(option);
    var dateFormatter = intl.getDateFormat(option);
    for (var i = 0; start <= end; i++) {
        date = Date.parse(dateParser(dateFormatter(start)));
        if (Math.random() > .5) {
            value += (Math.random() * 10 - 5);
        }
        else {
            value -= (Math.random() * 10 - 5);
        }
        if (value < 0) {
```

```

        value = getRandomInt(20, 40);
    }
    var point1 = { x: new Date(date), y: Math.round(value) };
    new Date(start.setDate(start.getDate() + 1));
    series1.push(point1);
}
return series1;
}
function getRandomInt(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
</script>

```

LOCAL-DATA.CS

```

public IActionResult Index()
{
    ViewBag.min = new DateTime(2009, 1, 1);
    ViewBag.max = new DateTime(2014, 1, 1);
    return View();
}

```

Customizing empty point

Specific color for empty point can be set by **Fill** property in **EmptyPointSettings**. Border for a empty point can be set by **Border** property.

CSHTML

```

@(Html.EJS().Chart("container").ChartArea(ca =>
ca.Border(ViewBag.border))
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
        .Name("Profit")
        .XName("xValue")
        .YName("yValue")
        .Marker(ViewBag.marker)
        .Width(2)
        .EmptyPointSettings(ViewBag.emptyPointSettings)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.Title("Product")
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .Interval(1)
    )
    .Title("Annual Product-Wise Profit Analysis")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").Render()
)

```

CUSTOM-EMPTYPOINT.CS

```

public IActionResult Index()
{
    List<EmptyData> chartData = new List<EmptyData>
    {
        new EmptyData{ month="Jan", sales= 35 },
        new EmptyData{ month="Feb", sales= 28 },
        new EmptyData{ month="Mar", sales=null },
        new EmptyData{ month="Apr", sales=32 },
        new EmptyData{ month="May", sales=40 },
        new EmptyData{ month="Jun", sales=32 },
        new EmptyData{ month="Jul", sales=35 },
        new EmptyData{ month="Aug", sales=null },
        new EmptyData{ month="Sep", sales=38 },
        new EmptyData{ month="Oct", sales=30 },
        new EmptyData{ month="Nov", sales=25 },
        new EmptyData{ month="Dec", sales=32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class EmptyData
{
    public string month;
    public Nullable<double> sales;
}

```

Chart Dimensions

Size for Container

Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Minimum(2)
.Maximum(5)
.Interval(2))
.Title("Olympic Medal Counts - RIO")
.Width("650")
.Height("350")
.Render()

```

SIZE.CS

```

public IActionResult Index()

```



```

    {
        List<ColumnChartData> chartData = new List<ColumnChartData>
        {
            new ColumnChartData { x= "USA", yValue= 46 },
            new ColumnChartData { x= "GBR", yValue= 27 },
            new ColumnChartData { x= "CHN", yValue= 26 },
            new ColumnChartData { x= "UK", yValue= 26 },
            new ColumnChartData { x= "AUS", yValue= 26 },
            new ColumnChartData { x= "IND", yValue= 26 },
            new ColumnChartData { x= "DEN", yValue= 26 },
            new ColumnChartData { x= "MEX", yValue= 26 },
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class ColumnChartData
    {
        public string x;
        public double yValue;
    }

```

Size for Chart

You can also set size for chart directly through [Width](#) and [Height](#) properties.

<!-- markdownlint-disable MD036 -->

In Pixel

<!-- markdownlint-disable MD036 -->

You can set the size of chart in pixel as demonstrated below.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Minimum(2)
.Maximum(5)
.Interval(2)).
Title("Olympic Medal Counts -
RIO").Width("650px").Height("350px").Render()

```

PIXEL.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>

```

```

        {
            new ColumnChartData { x= "USA", yValue= 46 },
            new ColumnChartData { x= "GBR", yValue= 27 },
            new ColumnChartData { x= "CHN", yValue= 26 },
            new ColumnChartData { x= "UK", yValue= 26 },
            new ColumnChartData { x= "AUS", yValue= 26 },
            new ColumnChartData { x= "IND", yValue= 26 },
            new ColumnChartData { x= "DEN", yValue= 26 },
            new ColumnChartData { x= "MEX", yValue= 26 },
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class ColumnChartData
    {
        public string x;
        public double yValue;
    }

```

In Percentage

By setting value in percentage, chart gets its dimension with respect to its container. For example, when the height is '50%', chart renders to half of the container height.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Minimum(2)
.Maximum(5)
.Interval(2)).
Title("Olympic Medal Counts - RIO").Width("80%").Height("90%").Render()

```

PERCENTAGE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
    }

```

```

        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
}

```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Category Axis

<!-- markdownlint-disable MD036 -->

Category axis are used to represent, the string values instead of numbers.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()

```

COLUMN.CS

```

//datasource for column chart
public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 23 },
        new ColumnChartData { x= "AUS", yValue= 16 },
        new ColumnChartData { x= "IND", yValue= 36 },
        new ColumnChartData { x= "DEN", yValue= 12 },
        new ColumnChartData { x= "MEX", yValue= 20 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
}

```

```

        public double yValue;
    }

```

<!-- markdownlint-disable MD036 -->

Labels Placement

<!-- markdownlint-disable MD036 -->

By default, category labels are placed between the ticks in an axis, this can also be placed on ticks using [LabelPlacement](#) property.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)).
Title("Olympic Medal Counts - RIO").Render()

```

PLACEMENT.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 23 },
        new ColumnChartData { x= "AUS", yValue= 16 },
        new ColumnChartData { x= "IND", yValue= 36 },
        new ColumnChartData { x= "DEN", yValue= 12 },
        new ColumnChartData { x= "MEX", yValue= 20 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
}

```

Range

Range of the category axis can be customized using [Minimum](#), [Maximum](#) and [Interval](#) property of the axis.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Minimum(2)
.Maximum(5)
.Interval(2)).
Title("Olympic Medal Counts - RIO").Render()
```

RANGE.CS

```
public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 23 },
        new ColumnChartData { x= "AUS", yValue= 16 },
        new ColumnChartData { x= "IND", yValue= 36 },
        new ColumnChartData { x= "DEN", yValue= 12 },
        new ColumnChartData { x= "MEX", yValue= 20 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
}
```

Indexed category axis

Category axis also can be rendered based on the index values of data source. This can be achieved by defining the `IsIndexed` property to `true` in the axis.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
```

```

        .Add();
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        XName("x").
        YName("yValue").
        DataSource(ViewBag.dataSource1).
        Name("Silver").
        .Add();
    }).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.IsIndexed(true)).
    Title("Olympic Medal Counts - RIO").Render()

```

INDEX.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ x= "Myanmar", y= 7.3 },
        new ColumnChartData{ x= "India", y= 7.9 },
        new ColumnChartData{ x= "Bangladesh", y= 6.8 },
        new ColumnChartData{ x= "Cambodia", y=7.0 },
        new ColumnChartData{ x= "China", y= 6.9 }
    };
    ViewBag.dataSource = chartData;
    List<ColumnChartData> chartData1 = new List<ColumnChartData>
    {
        new ColumnChartData{ x= "Poland", y=2.7 },
        new ColumnChartData{ x= "Australia", y=2.5 },
        new ColumnChartData{ x= "Singapore", y=2.0 },
        new ColumnChartData{ x= "Canada", y=1.4 },
        new ColumnChartData{ x= "Germany", y=1.8 }
    };
    ViewBag.dataSource1 = chartData1;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double y;
}

```

<!-- markdownlint-disable MD036 -->

Numeric Axis

You can use numeric [Axis](#) to represent numeric values of data in chart. By default, the `ValueType` of an axis is [Double](#).

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional)
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

DOUBLE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Range

Range for an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using [Minimum](#), [Maximum](#) and [Interval](#) property of the axis.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional)
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

RANGE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Range Padding

Padding can be applied to the minimum and maximum extremes of the axis range by using the [RangePadding](#) property. Numeric axis supports following types of padding.

- None
- Round
- Additional
- Normal
- Auto

Numeric - None

When the [RangePadding](#) is set to **None**, minimum and maximum of an axis is based on the data.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>

```



```

{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

NONE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Numeric - Round

When the [RangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value divisible by interval. For example, when the minimum is 3.5 and the interval is 1, then the minimum will be rounded to 3.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xVal

```

```

ue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Round)
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

ROUND.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Numeric - Additional

When the [RangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
}

```

```

    }).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional)
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

ADDITIONAL.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Numeric - Normal

When the [RangePadding](#) is set to **Normal**, padding is applied to the axis based on default range calculation.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Normal)
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)

```

```

        ).Title("Inflation - Consumer Price").Tooltip(tt =>
        tt.Enable(true)).Load("load").Render()
    }
}

```

NORMAL.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Numeric - Auto

When the [RangePadding](#) is set to `Auto`, horizontal numeric axis takes None as padding calculation, while the vertical numeric axis takes Normal as padding calculation.

CSHTML

```

@Html.EJS().Charts("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Auto)
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()
}
}

```

AUTO.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

Label Format

Numeric Label Format

Numeric labels can be formatted by using the [LabelFormat](#) property. Numeric labels supports all globalize format.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).LabelFormat("n1")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

LABEL-FORMAT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>

```

```

    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

GroupingSeparator

To separate groups of thousands, use [UseGroupingSeparator](#) property in chart.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{

```

```
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("x")
.YName("y").DataSource(ViewBag.dataSource).Name("Germany").Add();
}).Title("Inflation - Consumer
Price").UseGroupingSeparator(true).Render()
```

GROUP.CS

```
public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ x= "USA", y=50 },
        new ColumnChartData{ x="China", y=40 },
        new ColumnChartData{ x= "Japan", y=70 },
        new ColumnChartData{ x= "Australia", y=60},
        new ColumnChartData{ x= "France", y=50 },
        new ColumnChartData{ x= "Germany", y=40 },
        new ColumnChartData{ x= "Italy", y=40 },
        new ColumnChartData{ x= "Sweden", y=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double y;
}
```

Custom Label Format

Axis also supports custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

CSHTML

```
@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xVal
ue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).N
ame("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xVal
ue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).
Name("England").Add();
}).PrimaryXAxis(px =>
px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).LabelFor
mat("{value}K")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadd
ing.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()
```

CUSTOM.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { xValue = 10, yValue = 21, yValue1 = 28 },
        new ChartData { xValue = 20, yValue = 24, yValue1 = 44 },
        new ChartData { xValue = 30, yValue = 36, yValue1 = 48 },
        new ChartData { xValue = 40, yValue = 38, yValue1 = 50 },
        new ChartData { xValue = 50, yValue = 54, yValue1 = 66 },
        new ChartData { xValue = 60, yValue = 57, yValue1 = 78 },
        new ChartData { xValue = 70, yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}

```

<!-- markdownlint-disable MD036 -->

DateTime and DateTimeCategory Axis**DateTime Axis**

Date time axis uses date time scale and displays the date time values as axis labels in the specified format.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()

```

DATETIME.CS

```

public ActionResult Index()
{

```



```

        List<LineChartData> chartData = new List<LineChartData>
        {
            new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
            new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
            new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
            new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
        };
        ViewBag.dataSource = chartData;
        return View();
    }

    public class LineChartData
    {
        public DateTime xValue;
        public double yValue;
        public double yValue1;
    }

```

DateTimeCategory Axis

Date-time category axis is used to display the date-time values with non-linear intervals. For example, the business days alone have been depicted in a week here.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTimeCategory).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()

```

DATECATEGORY.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData{ x= new DateTime(2017, 11, 20), y= 21 },
        new LineChartData{ x= new DateTime(2017, 11, 21), y= 24 },
        new LineChartData{ x= new DateTime(2017, 11, 22), y= 24 },
        new LineChartData{ x= new DateTime(2017, 11, 26), y= 70 },
        new LineChartData{ x= new DateTime(2017, 11, 27), y= 75 },
    }
}

```

```

        new LineChartData{ x= new DateTime(2018, 0, 2), y= 82 },
        new LineChartData{ x= new DateTime(2018, 0, 3), y= 53 },
        new LineChartData{ x= new DateTime(2018, 0, 4), y= 54 },
        new LineChartData{ x= new DateTime(2018, 0, 5), y= 53 },
        new LineChartData{ x= new DateTime(2018, 0, 8), y= 45 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class LineChartData
{
    public DateTime x;
    public double y;
}

```

Range

Range for an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using [Minimum](#), [Maximum](#) and [Interval](#) property of the axis.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y").Minimum(new DateTime(2005, 1, 1)).Maximum(new DateTime(2010, 1, 1))
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()

```

RANGE.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

```

```
public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}
```

Interval Customization

Date time intervals can be customized by using the [Interval](#) and [IntervalType](#) properties of the axis. For example, when you set interval as 2 and intervalType as years, it considers 2 years as interval. DateTime axis supports following interval types,

- Auto
- Years
- Months
- Days
- Hours
- Minutes
- Seconds

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y").IntervalType("Months")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()
```

INTERVAL.CS

```
public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
    ViewBag.dataSource = chartData;
}
```

```

        return View();
    }
    public class LineChartData
    {
        public DateTime xValue;
        public double yValue;
        public double yValue1;
    }

```

Applying Padding to the Range

Padding can be applied to the minimum and maximum extremes of the range by using the [RangePadding](#) property. Date time axis supports the following types of padding,

- None
- Round
- Additional

DateTime - None

When the [RangePadding](#) is set to **None**, minimum and maximum of the axis is based on the data.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()

```

NONE.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
}

```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class LineChartData
    {
        public DateTime xValue;
        public double yValue;
        public double yValue1;
    }

```

DateTime - Round

When the [RangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value divisible by interval. For example, when the minimum is 15th Jan, interval is 1 and the interval type is 'month', then the axis minimum will be Jan 1st.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
})
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
    ).PrimaryYAxis(py =>
        py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Round).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Load("load").Render()

```

ROUND.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class LineChartData
{
    public DateTime xValue;
    public double yValue;
}

```

```

        public double yValue1;
    }

```

DateTime - Additional

When the [RangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Load("load").Render()

```

ADDITIONAL.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

Label Format

You can format and parse the date to all globalize format using [LabelFormat](#) property in an axis.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("yMd")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()
```

LABEL-FORMAT.CS

```
public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}
```

The following table describes the result of applying some common date time formats to the labelFormat property

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	EEEE	Monday	The Date is displayed in day format
new Date(2000, 03, 10)	yMd	04/10/2000	The Date is displayed in month/date/year format

new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed
new Date(2000, 03, 10)	hm	12:00 AM	Time of the date value is displayed as label
new Date(2000, 03, 10)	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format

<!-- markdownlint-disable MD033 -->

Custom Label Format

Axis also supports custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()
```

CUSTOM.CS

```
public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
```



```

        public double yValue;
        public double yValue1;
    }

```

Logarithmic Axis

<!-- markdownlint-disable MD033 -->

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numeric values in both lower order of magnitude (eg: $10^{⁻⁶}$) and higher order of magnitude (eg: $10^{⁶}$).

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(20)
).Minimum(0).Maximum(100)
).Render()

```

LOG.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data { xValue = new DateTime(2005, 01, 01), yValue = 100, yValue1 = 28 },
        new Data { xValue = new DateTime(2006, 01, 01), yValue = 200, yValue1 = 44 },
        new Data { xValue = new DateTime(2007, 01, 01), yValue = 500, yValue1 = 48 },
        new Data { xValue = new DateTime(2008, 01, 01), yValue = 1000, yValue1 = 50 },
        new Data { xValue = new DateTime(2009, 01, 01), yValue = 8000, yValue1 = 66 },
        new Data { xValue = new DateTime(2010, 01, 01), yValue = 90000, yValue1 = 78 },
        new Data { xValue = new DateTime(2011, 01, 01), yValue = 99000, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

```
}

```

Range

Range of an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using [Minimum](#), [Maximum](#) and [Interval](#) property of the axis.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Minimum(10).Maximum(1000)
).Render()
```

RANGE.CS

```
public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data { xValue = new DateTime(2005, 01, 01), yValue = 100, yValue1 = 28 },
        new Data { xValue = new DateTime(2006, 01, 01), yValue = 200, yValue1 = 44 },
        new Data { xValue = new DateTime(2007, 01, 01), yValue = 500, yValue1 = 48 },
        new Data { xValue = new DateTime(2008, 01, 01), yValue = 1000, yValue1 = 50 },
        new Data { xValue = new DateTime(2009, 01, 01), yValue = 8000, yValue1 = 66 },
        new Data { xValue = new DateTime(2010, 01, 01), yValue = 90000, yValue1 = 78 },
        new Data { xValue = new DateTime(2011, 01, 01), yValue = 99000, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}
```

Logarithmic Base

Logarithmic base can be customized by using the [LogBase](#) property of the axis. For example when the logBase is 5, the axis values follows 5^{-2} , 5^{-1} , 5^0 , 5^1 , 5^2 etc.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
    ).PrimaryYAxis(py =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).LabelFormat("{value}%")
    ).LogBase(8)
    ).Render()
```

BASE.CS

```
public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data { xValue = new DateTime(2005, 01, 01), yValue = 100, yValue1 = 28 },
        new Data { xValue = new DateTime(2006, 01, 01), yValue = 200, yValue1 = 44 },
        new Data { xValue = new DateTime(2007, 01, 01), yValue = 500, yValue1 = 48 },
        new Data { xValue = new DateTime(2008, 01, 01), yValue = 1000, yValue1 = 50 },
        new Data { xValue = new DateTime(2009, 01, 01), yValue = 8000, yValue1 = 66 },
        new Data { xValue = new DateTime(2010, 01, 01), yValue = 90000, yValue1 = 78 },
        new Data { xValue = new DateTime(2011, 01, 01), yValue = 99000, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}
```

Logarithmic Interval

Logarithmic axis interval can be customized by using the [Interval](#) property of the axis. When the logarithmic base is 10 and logarithmic interval is 2, then the axis labels are placed at an interval of 10^2 . The default value of the interval is 1.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
    ).PrimaryYAxis(py =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).Interval(2)
    ).Render()
```

INTERVAL.CS

```
public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data { xValue = new DateTime(2005, 01, 01), yValue = 100, yValue1 = 28 },
        new Data { xValue = new DateTime(2006, 01, 01), yValue = 200, yValue1 = 44 },
        new Data { xValue = new DateTime(2007, 01, 01), yValue = 500, yValue1 = 48 },
        new Data { xValue = new DateTime(2008, 01, 01), yValue = 1000, yValue1 = 50 },
        new Data { xValue = new DateTime(2009, 01, 01), yValue = 8000, yValue1 = 66 },
        new Data { xValue = new DateTime(2010, 01, 01), yValue = 90000, yValue1 = 78 },
        new Data { xValue = new DateTime(2011, 01, 01), yValue = 99000, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}
```

Axis Labels

Smart Axis Labels

When the axis labels overlap with each other, you can use [LabelIntersectAction](#) property in the axis, to place them smartly.

When setting [LabelIntersectAction](#) as **Hide**

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue")
    .DataSource(ViewBag.dataSource)
    .Name("Germany").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue1")
    .DataSource(ViewBag.dataSource)
    .Name("England").Add();
}).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .MajorGridLines(ViewBag.majorGridLines)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Hides)
    .LabelFormat("y")
).PrimaryYAxis(py =>
    py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(ViewBag.majorTickLines)
    .MinorTickLines(ViewBag.minorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Render()

```

HIDE.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{

```

```

        public string x;
        public double y;
        public double y1;
    }

```

When setting **LabelIntersectAction** as **Rotate45**

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue")
    .DataSource(ViewBag.dataSource)
    .Name("Germany").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue1")
    .DataSource(ViewBag.dataSource)
    .Name("England").Add();
}).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .MajorGridLines(ViewBag.majorGridLines)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Hides)
    .LabelFormat("y")
).PrimaryYAxis(py =>
    py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(ViewBag.majorTickLines)
    .MinorTickLines(ViewBag.minorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Render()

```

ROTATE45.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
    }
}

```

```

        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
    public double y1;
}

```

When setting **LabelIntersectAction** as **Rotate90**

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue")
    .DataSource(ViewBag.dataSource)
    .Name("Germany").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue1")
    .DataSource(ViewBag.dataSource)
    .Name("England").Add();
}).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .MajorGridLines(ViewBag.majorGridLines)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Hides)
    .LabelFormat("y")
).PrimaryYAxis(py =>
    py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(ViewBag.majorTickLines)
    .MinorTickLines(ViewBag.minorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Render()

```

ROTATE90.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {

```

```

new AxisLabelData { x= "South Korea", y= 39.4 },
new AxisLabelData { x= "India", y= 61.3 },
new AxisLabelData { x= "Pakistan", y= 20.4 },
new AxisLabelData { x= "Germany", y= 65.1 },
new AxisLabelData { x= "Australia", y= 15.8 },
new AxisLabelData { x= "Italy", y= 29.2 },
new AxisLabelData { x= "United Kingdom", y= 44.6 },
new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
new AxisLabelData { x= "Russia", y= 40.8 },
new AxisLabelData { x= "Mexico", y= 31 },
new AxisLabelData { x= "Brazil", y= 75.9 },
new AxisLabelData { x= "China", y= 51.4 }
};
ViewBag.dataSource = chartData;
return View();
}
public class AxisLabelData
{
    public string x;
    public double y;
    public double y1;
}

```

Axis Labels Positioning

By default, the axis labels can be placed at **Outside** the axis line and this also can be placed at **Inside** the axis line using the **LabelPosition** property.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x")
    .YName("y")
    .Marker(ViewBag.marker)
    .DataSource(ViewBag.dataSource)
    .Name("Germany").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.MajorGridLines(ViewBag.majorGridLines)
.IsIndexed(true)
.LabelRotation(90)
.Interval(1)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90)
).PrimaryYAxis(py => py.Maximum(120)
.Minimum(0)
.Interval(30)
.MajorTickLines(ViewBag.majorTickLines)
.LineStyle(ViewBag.lineStyle)
).Title("Fruits and Vegetables - Season").ChartArea(area =>
area.Border(ViewBag.ChartBorder))
.LegendSettings(lg=>lg.Visible(false))
.Load("load")
.Render()
<script>

```



```

var load = function (args) {
    var selectedTheme = location.hash.split('/')[1];
    selectedTheme = selectedTheme ? selectedTheme : 'Material';
    args.chart.theme = (selectedTheme.charAt(0).toUpperCase() +
selectedTheme.slice(1));
    args.chart.width = ej.base.Browser.isDevice ? "100%" : "60%";
    args.chart.primaryXAxis.border = { type: "Rectangle", width: 1
};

    args.chart.primaryXAxis.multiLevelLabels =
(ej.base.Browser.isDevice ? ([
        {
            border: { type: 'Rectangle' },
            categories: [
                { start: -0.5, end: 2.5, text: 'In Season', },
                { start: 2.5, end: 5.5, text: 'Out of Season', },
                { start: 5.5, end: 7.5, text: 'In Season', },
                { start: 7.5, end: 9.5, text: 'Out of Season', },
            ]
        }, {
            border: { type: 'Rectangle' },
            textStyle: { fontWeight: 'Bold' },
            categories: [
                { start: -0.5, end: 5.5, text: 'Fruits', },
                { start: 5.5, end: 9.5, text: 'Vegetables', },
            ]
        }) : [
        {
            border: { type: 'Rectangle' },
            categories: [
                { start: -0.5, end: 0.5, text: 'Seedless', },
                { start: 0.5, end: 2.5, text: 'Seeded', },
                { start: 2.5, end: 3.5, text: 'Seedless', },
                { start: 3.5, end: 5.5, text: 'Seeded', },
                { start: 5.5, end: 6.5, text: 'Seedless', },
                { start: 6.5, end: 7.5, text: 'Seeded', },
                { start: 7.5, end: 8.5, text: 'Seedless', },
                { start: 8.5, end: 9.5, text: 'Seeded', },
            ]
        }, {
            border: { type: 'Rectangle' },
            categories: [
                { start: -0.5, end: 2.5, text: 'In Season', },
                { start: 2.5, end: 5.5, text: 'Out of Season', },
            ]
        }, {
            border: { type: 'Rectangle' },
            categories: [
                { start: 5.5, end: 7.5, text: 'In Season', },
                { start: 7.5, end: 9.5, text: 'Out of Season', },
            ]
        }, {
            border: { type: 'Rectangle' },
            textStyle: { fontWeight: 'Bold' },
            categories: [
                { start: -0.5, end: 5.5, text: 'Fruits', },
                { start: 5.5, end: 9.5, text: 'Vegetables', },
            ]
        })
    ]);
};

```

```
</script>
```

POSITION.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
    public double y1;
}
```

Multilevel Labels

Any number of levels of labels can be added to an axis using the `MultiLevelLabels` property. This property can be configured using the following properties:

- Categories
- Overflow
- Alignment
- Text style
- Border

Categories

Using the categories property, you can configure the `Start`, `End`, `Text`, and `MaximumTextWidth` of multilevel labels.

CSHTML

```
@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
})
```

```

        }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).LabelFormat("y")
        ).PrimaryYAxis(py => py.LabelFormat("{value}%").
        ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

CATEGORY.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "United States", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Overflow

Using the **Overflow** property, you can **Trim** or **Wrap** the multilevel labels.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
        }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
        ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.mino

```

```

rTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
)
).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

OVERFLOW.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "United States", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Alignment

The **Alignment** property provides option to position the multilevel labels at **Far**, **Center**, or **Near**.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();

}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
)

```

```

        ).Title("Inflation - Consumer Price").Tooltip(tt =>
        tt.Enable(true)).Load("load").Render()
    }
}

```

ALIGNEMENT.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "United States", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Text customization

The `TextStyle` property of multilevel labels provides options to customize the `Size`, `Color`, `FontFamily`, `FontWeight`, `FontStyle`, `Opacity`, `TextAlignment` and `TextOverflow`.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
)

```

```

        ).Title("Inflation - Consumer Price").Tooltip(tt =>
        tt.Enable(true)).Load("load").Render()
    }
}

```

TEXT-CUSTOM.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "United States", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Border customization

Using the **Border** property, you can customize the **Width**, **Color**, and **Type**. The **Type** of border are **Rectangle**, **Brace**, **WithoutBorder**, **WithoutTopBorder**, **WithoutTopandBottomBorder** and **CurlyBrace**.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
})}.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
)

```

```

        ).Title("Inflation - Consumer Price").Tooltip(tt =>
        tt.Enable(true)).Load("load").Render()
    }
}

```

BORDER-CUSTOM.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Chart;
namespace EJ2MVCSampleBrowser.Controllers.Chart
{
    public partial class ChartController : Controller
    {
        // GET: MultiLevelLabels
        public ActionResult MultiLevelLabels()
        {
            List<MultiLevelLabelsData> chartData = new
            List<MultiLevelLabelsData>
            {
                new MultiLevelLabelsData { x = "Grapes", y = 28 },
                new MultiLevelLabelsData { x = "Apples", y = 87 },
                new MultiLevelLabelsData { x = "Pears", y = 42 },
                new MultiLevelLabelsData { x = "Grapes", y = 13 },
                new MultiLevelLabelsData { x = "Apples", y = 13 },
                new MultiLevelLabelsData { x = "Pears", y = 10 },
                new MultiLevelLabelsData { x = "Tomato", y = 31 },
                new MultiLevelLabelsData { x = "Potato", y = 96 },
                new MultiLevelLabelsData { x = "Cucumber", y = 41 },
                new MultiLevelLabelsData { x = "Onion", y = 59 }
            };
            ViewBag.dataSource = chartData;
            return View();
        }
        public class MultiLevelLabelsData
        {
            public string x;
            public double y;
        }
    }
}

```

Edge Label Placement

Labels with long text at the edges of an axis may appear partially in the chart. To avoid this, use [EdgeLabelPlacement](#) property in axis, which moves the label inside the chart area for better appearance or hides it.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
}
)

```

```

        .XName("xValue")
        .YName("yValue")
        .DataSource(ViewBag.dataSource)
        .Name("Germany").Add();
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
.Width(2)
.XName("xValue")
.YName("yValue1")
.DataSource(ViewBag.dataSource)
.Name("England").Add();
}).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .MajorGridLines(ViewBag.majorGridLines)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Hide)
.LabelFormat("y")
).PrimaryYAxis(py =>
    py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(ViewBag.majorTickLines)
    .MinorTickLines(ViewBag.minorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Render()

```

EDGE.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "Russia", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "United States", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Labels Customization

The [LabelStyle](#) property of an axis provides options to customize the **Color**, **Font-family**, **Font-size** and **Font-weight** of the axis labels.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue")
    .DataSource(ViewBag.dataSource)
    .Name("Germany").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue1")
    .DataSource(ViewBag.dataSource)
    .Name("England").Add();
}).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .MajorGridLines(ViewBag.majorGridLines)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Hides)
    .LabelFormat("y")
    .LabelStyle("ViewBag.label")
).PrimaryYAxis(py =>
    py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(ViewBag.majorTickLines)
    .MinorTickLines(ViewBag.minorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Render()

```

LABELS-CUSTOM.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "Russia", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "United States", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;

```

```

        public double gold;
    }

```

Customizing Specific Point

You can customize the specific text in the axis labels using `AxisLabelRender` event.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue")
    .DataSource(ViewBag.dataSource)
    .Name("Germany").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Width(2)
    .XName("xValue")
    .YName("yValue1")
    .DataSource(ViewBag.dataSource)
    .Name("England").Add();
}).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .MajorGridLines(ViewBag.majorGridLines)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Hides)
    .LabelFormat("y")
).PrimaryYAxis(py =>
    py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(ViewBag.majorTickLines)
    .MinorTickLines(ViewBag.minorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer
Price").AxisLabelRender("axisLabel").Render()
<script>
function axislabel(args) {
    if(args.text === 'France') {
        args.labelStyle.color = 'Red';
    }
}
</script>

```

CUSTOM-POINT.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "Russia", gold=50 },
    }
}

```

```

        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "United States", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Trim using maximum label width

You can trim the label using [EnableTrim](#) property and width of the labels can also be customized using [MaximumLabelWidth](#) property in the axis, the value maximum label width is 34 by default.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.TooltipMappingName("country").Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar)
        .XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).PrimaryXAxis(px =>
px.EnableTrim(true).Title("Country").ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Render()

```

LABELS-TRIM.CS

```

public ActionResult Index()
{
    List<CategoryData> chartData = new List<ColumnChartData>
    {
        new CategoryData { x = "Germany", y = 72, country = "GER: 72"},
        new CategoryData { x = "Russia", y = 103.1, country = "RUS: 103.1" },
        new CategoryData { x = "Brazil", y = 139.1, country = "BRZ: 139.1" },
        new CategoryData { x = "India", y = 462.1, country = "IND: 462.1" },
        new CategoryData { x = "China", y = 721.4, country = "CHN: 721.4" },
        new CategoryData { x = "United States<br>Of America", y = 286.9, country = "USA: 286.9" },
        new CategoryData { x = "Great Britain", y = 115.1, country = "GBR: 115.1" },
    }
}

```

```

        new CategoryData { x = "Nigeria", y = 97.2, country = "NGR:
97.2" },
    };
    ViewBag.dataSource = chartData;
    ViewBag.font = new
    {
        fontWeight = "600",
        color = "#ffffff"
    };
    return View();
}
public class CategoryData
{
    public string x;
    public double y;
    public string country;
}

```

Line break support

Line break feature used to customize the long axis label text into multiple lines by using `
` tag. Refer the below example in that dataSource x value contains long text, it breaks into two lines by using `
` tag.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.TooltipMappingName("country").Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar)
        .XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).PrimaryXAxis(px =>
px.EnableTrim(true).Title("Country").ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Render()

```

LINE-BREAK.CS

```

public ActionResult Index()
{
    List<CategoryData> chartData = new List<ColumnChartData>
    {
        new CategoryData { x = "Germany", y = 72, country = "GER:
72"},
        new CategoryData { x = "Russia", y = 103.1, country = "RUS:
103.1" },
        new CategoryData { x = "Brazil", y = 139.1, country = "BRZ:
139.1" },
        new CategoryData { x = "India", y = 462.1, country = "IND:
462.1" },
        new CategoryData { x = "China", y = 721.4, country = "CHN:
721.4" },
        new CategoryData { x = "United States<br>Of America", y =
286.9, country = "USA: 286.9" },
    }
}

```

```

        new CategoryData { x = "Great Britain", y = 115.1, country =
"GBR: 115.1" },
        new CategoryData { x = "Nigeria", y = 97.2, country = "NGR:
97.2" },
    };
    ViewBag.dataSource = chartData;
    ViewBag.font = new
    {
        fontWeight = "600",
        color = "#ffffff"
    };
    return View();
}
public class CategoryData
{
    public string x;
    public double y;
    public string country;
}

```

Maximum Labels

MaximumLabels property is set, then the labels will be rendered based on the count in the property per 100 pixel. If you have set range (minimum, maximum, interval) and maximumLabels, then the priority goes to range only. If you haven't set the range, then we have considered priority to maximumLabels property.

CSHTML

```

@Html.EJS().Chart("container").PrimaryXAxis(px => px.Title("Years").
ValueType(Syncfusion.EJ2.Charts.ValueType.Double).MaximumLabels(1)).Series(s
eries =>
    {
        series.Type(
            Syncfusion.EJ2.Charts.ChartSeriesType.Line
        ).DataSource("series1").Name("Product
X").XName("x").YName("y").Add();
    }
).Crosshair(cr => cr.Enable(true)).ZoomSettings(z =>
z.EnableMouseWheelZooming(true).EnablePinchZooming(true).EnableSelectionZoom
ing(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X).EnableScrollbar(true)).Prim
aryYAxis(py => py.Title("Profit ($)").Title("Sales History of Product
X")).Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 50; i++) {
        if (Math.random() > .5) {
            value += Math.random();
        } else {
            value -= Math.random();
        }
    }
    point1 = { x: i, y: value.toFixed(1) };

```

```

        series1.push(point1);
    }
</script>

```

MAX-LABEL.CS

```

public ActionResult Index()
{
    return View();
}

```

Axis Customization in ASP.NET MVC Chart Component

Axis Crossing

An axis can be positioned in the chart area using **CrossesAt** and **CrossesInAxis** properties. The **CrossesAt** property specifies the values (datetime, numeric, or logarithmic) at which the axis line has to be intersected with the vertical axis or vice-versa, and the **CrossesInAxis** property specifies the axis name with which the axis line has to be crossed.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).
crossesAt(15)).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).
crossesAt(5)).
    Title("Olympic Medal Counts - RIO").Render()

```

AXIS-CROSS.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class ColumnChartData
    {
        public string x;
        public double yValue;
    }

```

Title

You can add a title to the axis using [Title](#) property to provide quick information to the user about the data plotted in the axis.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).TitleStyle
e(ViewBag.titleStyle)).
    Title("Olympic Medal Counts - RIO").Render()

```

TITLE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string x;
    public double yValue;
}

```

Title Rotation

By using the [titleRotation](#) property, you can rotate the axis title from 0 to 360 degree.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).TitleStyle(
ViewBag.titleStyle).TitleRotation(90)).
Title("Olympic Medal Counts - RIO").Render()
```

TITLEROTATION.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
}
```

Tick Lines Customization

You can customize the **Width**, **Color** and **Size** of the minor and major tick lines, using [MajorTickLines](#) and [MinorTickLines](#) properties in the axis.

CSHTML

```
@Html.EJS().Chart("container").PrimaryXAxis(px => px.Title("Years").
    ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .MajorTickLines(mg =>
mg.Color("blue").Width(2))).Series(series =>
{
    series.Type(
```



```

        Syncfusion.EJ2.Charts.ChartSeriesType.Column
    ).DataSource(ViewBag.dataSource).Name("Product
X").XName("x").YName("yValue").Add();
    }
    ).PrimaryYAxis(py => py.Title("Profit ($)")
    .MajorTickLines(mg =>
mg.Color("blue").Width(2))).Title("Sales History of Product X").Render()

```

TICK.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "John", yValue= 10000, yValue1=37,
yValue2=38 },
        new ColumnChartData { x= "Jake", yValue= 12000, yValue1=23,
yValue2=17 },
        new ColumnChartData { x= "Peter", yValue= 18000, yValue1=18,
yValue2=26 },
        new ColumnChartData { x= "James", yValue= 11000, yValue1=18,
yValue2=26 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
    public double yValue1;
    public double yValue2;
}

```

Grid Lines Customization

You can customize the **Width**, **Color** and **DashArray** of the minor and major grid lines, using [MajorGridLines](#) and [MinorGridLines](#) properties in the axis.

CSHTML

```

@Html.EJS().Chart("container").PrimaryXAxis(px => px.Title("Years").
    ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .MajorGridLines(mg =>
mg.Color("blue").Width(2))).Series(series =>
{
    series.Type(
        Syncfusion.EJ2.Charts.ChartSeriesType.Column
    ).DataSource(ViewBag.dataSource).Name("Product
X").XName("x").YName("yValue").Add();
    }
    ).PrimaryYAxis(py => py.Title("Profit ($)")
    .MajorGridLines(mg =>
mg.Color("blue").Width(2))).Title("Sales History of Product X").Render()

```

GRID.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "John", yValue= 10000, yValue1=37,
yValue2=38 },
        new ColumnChartData { x= "Jake", yValue= 12000, yValue1=23,
yValue2=17 },
        new ColumnChartData { x= "Peter", yValue= 18000, yValue1=18,
yValue2=26 },
        new ColumnChartData { x= "James", yValue= 11000, yValue1=18,
yValue2=26 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
    public double yValue1;
    public double yValue2;
}

```

Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the chart. Series can be associated with this axis, by mapping with axis's unique name.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("y").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).YAxisName("yAxis").XName("x").Marker(ViewBag.marker).YName("y1").DataSource(ViewBag.dataSource).Name("Japan").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Interval(1).LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90)
).Axes(ax =>
{
    ax.OpposedPosition(true).RowIndex(0).Minimum(24).Maximum(36).Interval(2).Name("yAxis").LabelFormat("{value}°C").Add();
}
)
).Title("Weather Condition JPN vs DEU").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Tooltip(tt =>
tt.Enable(true)).LegendSettings(lg=>lg.Visible(false)).Render()

```

MULTIPLE.CS

```

public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
    {
        new MultipleAxesChartData { x = "Sun", y = 35, y1 = 30 },
        new MultipleAxesChartData { x = "Mon", y = 40, y1 = 28 },
        new MultipleAxesChartData { x = "Tue", y = 80, y1 = 29 },
        new MultipleAxesChartData { x = "Wed", y = 70, y1 = 30 },
        new MultipleAxesChartData { x = "Thu", y = 65, y1 = 33 },
        new MultipleAxesChartData { x = "Fri", y = 55, y1 = 32 },
        new MultipleAxesChartData { x = "Sat", y = 50, y1 = 34 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class MultipleAxesChartData
{
    public string x;
    public double y;
    public double y1;
}

```

Inversed Axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner set this property [IsInversed](#) to true.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
Marker(ViewBag.Marker).
XName("x").
YName("yValue").
DataSource(ViewBag.dataSource).
Name("Gold").
Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).IsInverse
d(true)).
Title("Olympic Medal Counts - RIO").Render()

```

INVERSED.CS

```

public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>

```

```

        {
            new MultipleAxesChartData { x = "Sun", y = 35, y1 = 30 },
            new MultipleAxesChartData { x = "Mon", y = 40, y1 = 28 },
            new MultipleAxesChartData { x = "Tue", y = 80, y1 = 29 },
            new MultipleAxesChartData { x = "Wed", y = 70, y1 = 30 },
            new MultipleAxesChartData { x = "Thu", y = 65, y1 = 33 },
            new MultipleAxesChartData { x = "Fri", y = 55, y1 = 32 },
            new MultipleAxesChartData { x = "Sat", y = 50, y1 = 34 }
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class MultipleAxesChartData
    {
        public string x;
        public double y;
        public double y1;
    }
}

```

Opposed Position

<!-- markdownlint-disable MD012 -->

To place an axis opposite from its original position, set [OpposedPosition](#) property of the axis to true.

<!-- markdownlint-disable MD012 -->

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).OpposedPosition(true)).
    Title("Olympic Medal Counts - RIO").Render()

```

OPPOSED.CS

```

public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
    {
        new MultipleAxesChartData { x = "Sun", y = 35, y1 = 30 },
        new MultipleAxesChartData { x = "Mon", y = 40, y1 = 28 },
        new MultipleAxesChartData { x = "Tue", y = 80, y1 = 29 },
        new MultipleAxesChartData { x = "Wed", y = 70, y1 = 30 },
        new MultipleAxesChartData { x = "Thu", y = 65, y1 = 33 },
        new MultipleAxesChartData { x = "Fri", y = 55, y1 = 32 },
    }
}

```

```

        new MultipleAxesChartData { x = "Sat", y = 50, y1 = 34 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class MultipleAxesChartData
{
    public string x;
    public double y;
    public double y1;
}

```

<!-- markdownlint-disable MD036 -->

Strip lines

<!-- markdownlint-disable MD036 -->

EJ2 chart supports horizontal and vertical strip lines and customization of stripline in both orientation.

Horizontal Strip lines

You can create Horizontal stripline by adding the **Stripline** in the vertical axis and set **Visible** option to true. Striplines are rendered in the specified start to end range and you can add more than one stripline for an axis.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
StripLines(ViewBag.yAxisStripline).
Title("Olympic Medal Counts - RIO").Render()

```

HORIZONTAL.CS

```

public ActionResult Index()
{
    List<StripLineChartData> chartData = new
List<StripLineChartData>
{
    new StripLineChartData { x = "Sun", y = 28 },
    new StripLineChartData { x = "Mon", y = 27 },
    new StripLineChartData { x = "Tue", y = 33 },
    new StripLineChartData { x = "Wed", y = 36 },
    new StripLineChartData { x = "Thu", y = 28 },
    new StripLineChartData { x = "Fri", y = 30 },
    new StripLineChartData { x = "Sat", y = 31 }
}

```

```

    };
    ViewBag.dataSource = chartData;
    return View();
}
public class StripLineChartData
{
    public string x;
    public double y;
}

```

Vertical Striplines

You can create vertical stripline by adding the **Stripline** in the horizontal axis and set **Visible** option to true. Striplines are rendered in the specified start to end range and you can add more than one stripline for an axis.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    StripLines(ViewBag.xAxisStripline).
    Title("Olympic Medal Counts - RIO").Render()

```

VERTICAL.CS

```

public ActionResult Index()
{
    List<StripLineChartData> chartData = new
List<StripLineChartData>
    {
        new StripLineChartData { x = "Sun", y = 28 },
        new StripLineChartData { x = "Mon", y = 27 },
        new StripLineChartData { x = "Tue", y = 33 },
        new StripLineChartData { x = "Wed", y = 36 },
        new StripLineChartData { x = "Thu", y = 28 },
        new StripLineChartData { x = "Fri", y = 30 },
        new StripLineChartData { x = "Sat", y = 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class StripLineChartData
{
    public string x;
    public double y;
}

```

Customize the strip line

Starting value in specific strip line can be customized by **Start** property in strip line. Similarly, ending value is customized by **End**. It can be also set for starting from the corresponding origin of the axis by **StartFromOrigin**. Size of the strip line is customized by **Size**. Border for the stripline is customized by **Border**. Order of the strip line such that whether it should be rendered in behind or over the series elements is customized by **ZIndex**.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
StripLines(ViewBag.xAxisStripLine).
Title("Olympic Medal Counts - RIO").Render()
```

CUSTOM-STRIPLINE.CS

```
public ActionResult Index()
{
    List<StripLineChartData> chartData = new
List<StripLineChartData>
    {
        new StripLineChartData { x = "Sun", y = 28 },
        new StripLineChartData { x = "Mon", y = 27 },
        new StripLineChartData { x = "Tue", y = 33 },
        new StripLineChartData { x = "Wed", y = 36 },
        new StripLineChartData { x = "Thu", y = 28 },
        new StripLineChartData { x = "Fri", y = 30 },
        new StripLineChartData { x = "Sat", y = 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class StripLineChartData
{
    public string x;
    public double y;
}
```

Customize the stripline text

You can customize the text rendered in stripline by **TextStyle** property. Rotation of the strip line text can be changed by **Rotation** property. Horizontal and Vertical alignment of stripline text can be changed by **HorizontalAlignment** and **VerticalAlignment** property.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
StripLines(ViewBag.xAxisStripLine).
Title("Olympic Medal Counts - RIO").Render()
```

CUSTOM-STRIPTEXT.CS

```
public ActionResult Index()
{
    List<StripLineChartData> chartData = new
List<StripLineChartData>
    {
        new StripLineChartData { x = "Sun", y = 28 },
        new StripLineChartData { x = "Mon", y = 27 },
        new StripLineChartData { x = "Tue", y = 33 },
        new StripLineChartData { x = "Wed", y = 36 },
        new StripLineChartData { x = "Thu", y = 28 },
        new StripLineChartData { x = "Fri", y = 30 },
        new StripLineChartData { x = "Sat", y = 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class StripLineChartData
{
    public string x;
    public double y;
}
```

Dash Array

You can create dash array stripline by using **DashArray** property. The Striplines are rendered with specified dash array values.

CSHTML

```
@(Html.EJS().Chart("container").EnableAnimation(true).Series(series =>
{
```



```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Interval(1)).
    PrimaryYAxis(py => py.Interval(10).StripLines(sl => {
sl.Visible(true).Start(30).Size(2).SizeType(Syncfusion.EJ2.Charts.SizeType.Pixel).DashArray("10,5").Color("red").Add(); })
    .Render()
)

```

DASH.CS

```

public ActionResult Index()
{
    List<StripLineChartData> chartData = new
List<StripLineChartData>
    {
        new StripLineChartData { x = "Sun", y = 28 },
        new StripLineChartData { x = "Mon", y = 27 },
        new StripLineChartData { x = "Tue", y = 33 },
        new StripLineChartData { x = "Wed", y = 36 },
        new StripLineChartData { x = "Thu", y = 28 },
        new StripLineChartData { x = "Fri", y = 30 },
        new StripLineChartData { x = "Sat", y = 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class StripLineChartData
{
    public string x;
    public double y;
}

```

Recurrence Stripline

The strip lines to be drawn repeatedly at the regular intervals – this will be useful when you want to mark an event that occurs recursively along the timeline of the chart. Following properties are used to configure this feature.

- **IsRepeat** - It is used for enable / disable the recurrence strip line.
- **RepeatEvery** - It is used for mention the stripline interval.
- **RepeatUntil** - It specifies the end value at which point strip line has to stop repeating.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).

```

```

        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
    }).
        PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Interval(1).StripLines(s1 => {
    s1.Visible(true).IsRepeat(true).RepeatEvery(2).Start(1).Color("rgba(167,169,171, 0.3)").Size(1).Add(); })).
        PrimaryYAxis(py => py.Interval(10))
    .Render()
)

```

RECURRENCE.CS

```

public ActionResult Index()
{
    List<StripLineChartData> chartData = new
List<StripLineChartData>
    {
        new StripLineChartData { x = 1, y = 28 },
        new StripLineChartData { x = 2, y = 27 },
        new StripLineChartData { x = 3, y = 33 },
        new StripLineChartData { x = 4, y = 36 },
        new StripLineChartData { x = 5, y = 28 },
        new StripLineChartData { x = 6, y = 30 },
        new StripLineChartData { x = 7, y = 31 }
    };
    List<ChartStripLine> yaxisstriplines = new
List<ChartStripLine>();
    ChartStripLine stripy = new ChartStripLine();
    stripy.Start = 1;
    stripy.Size = 1;
    stripy.IsRepeat = true;
    stripy.RepeatEvery = 2;
    stripy.Color = "rgba(167,169,171, 0.3)";
    stripy.Visible = true;
    yaxisstriplines.Add(stripy);
    ViewBag.yAxis = yaxisstriplines;
    ViewBag.dataSource = chartData;
    return View();
}

public class StripLineChartData
{
    public double x;
    public double y;
}

```

Size Type

The **SizeType** property refers the size of the stripline. They are,

- **Auto**

- Pixel
- Years
- Months
- Days
- Hours
- Minutes
- Seconds

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        XName("xValue").
        YName("yValue").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
})).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).StripLines(sl => {
    sl.Visible(true).Start(new System.DateTime(2005, 01, 01)).Size(1).SizeType(Syncfusion.EJ2.Charts.SizeType.Years).Color("rgba(167, 169, 171, 0.3)").Add(); })).
    PrimaryYAxis(py => py.Interval(10))
    .Render()
)
```

SIZE-TYPE.CS

```
public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38 },
    };
    ViewBag.dataSource = chartData;
    List<ChartStripLine> yaxisstriplines = new
List<ChartStripLine>();
    ChartStripLine stripy = new ChartStripLine();
    stripy.Start = new DateTime(2005, 01, 01);
    stripy.Size = 1;
    stripy.SizeType = SizeType.Years;
    stripy.Color = "rgba(167,169,171, 0.3)";
    stripy.Visible = true;
    yaxisstriplines.Add(stripy);
}
```

```

        ViewBag.yAxis = yaxisstriplines;
        return View();
    }

    public class LineChartData
    {
        public System.DateTime xValue;
        public double yValue;
    }

```

Segment Stripline

You can create stripline in a particular region with respect to segment. You can enable the segment stripline using `IsSegmented` property. The start and end value of this type of stripline can be defined using `SegmentStart` and `SegmentEnd` properties.

- `IsSegmented` - It is used for enable the segment stripline.
- `SegmentStart` - Used to change the segment start value. Value correspond to associated axis.
- `SegmentEnd` - Used to change the segment end value. Value correspond to associated axis.
- `SegmentAxisName` - Used to specify the name of the associated axis.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
        XName("xValue").
        YName("yValue").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
}))
    PrimaryYAxis(py => py.Interval(10).StripLines(sl => {
sl.Visible(true).Start(20).End(40).IsSegmented(true).SegmentStart(2).Segment
End(4).Color("rgba(167,169,171, 0.3)").Add(); }))
    .Render()
)

```

SEGMENT.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = 1, yValue = 5 },
        new LineChartData { xValue = 2, yValue = 39 },
        new LineChartData { xValue = 3, yValue = 21 },
        new LineChartData { xValue = 4, yValue = 51 },
        new LineChartData { xValue = 5, yValue = 30 },
        new LineChartData { xValue = 6, yValue = 26 }
    };
    List<ChartStripLine> yaxisstriplines = new
List<ChartStripLine>();
    ChartStripLine stripy = new ChartStripLine();
    stripy.Start = 20;
}

```

```

        stripy.End = 40;
        stripy.IsSegmented = true;
        stripy.SegmentStart = 2;
        stripy.SegmentEnd = 4;
        stripy.Color = "rgba(167,169,171, 0.3)";
        stripy.Visible = true;
        yaxisstriplines.Add(stripy);
        ViewBag.yAxis = yaxisstriplines;
        ViewBag.dataSource = chartData;
        return View();
    }
    public class LineChartData
    {
        public double xValue;
        public double yValue;
    }

```

Multiple Panes

Chart area can be divided into multiple panes using [Rows](#) and [Columns](#).

Rows

To split the chart area vertically into number of rows, use [Rows](#) property of the chart.

- You can allocate space for each row by using the [Height](#) property. The value can be either in percentage or in pixel.
- To associate a vertical axis to a particular row, specify its index to [RowIndex](#) property of the axis.
- To customize each row's bottom line, use [Border](#) property.

CSHTML

```

@Html.EJS().Chart("container").Rows(rows =>
{
    rows.Height("50%").Add();
    rows.Height("50%").Add();
})
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("y").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).YAxisName("yAxis").XName("x").Marker(ViewBag.marker).YName("y1").DataSource(ViewBag.dataSource).Name("Japan").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Interval(1).LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90)
).Axes(ax =>
{
    ax.OpposedPosition(true).RowIndex(0).Minimum(24).Maximum(36).Interval(2).Name("yAxis").LabelFormat("{value} °C").Add();
}

```

```

    )
).Title("Weather Condition JPN vs DEU").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Tooltip(tt =>
tt.Enable(true)).LegendSettings(lg=>lg.Visible(false)).Render()

```

ROW.CS

```

public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
{
    new MultipleAxesChartData{ x= "Jan", y= 15, y1= 33 },
    new MultipleAxesChartData{ x= "Feb", y= 20, y1= 31 },
    new MultipleAxesChartData{ x= "Mar", y= 35, y1= 30 },
    new MultipleAxesChartData{ x= "Apr", y= 40, y1= 28 },
    new MultipleAxesChartData{ x= "May", y= 80, y1= 29 },
    new MultipleAxesChartData{ x= "Jun", y= 70, y1= 30 },
    new MultipleAxesChartData{ x= "Jul", y= 65, y1= 33 },
    new MultipleAxesChartData{ x= "Aug", y= 55, y1= 32 },
    new MultipleAxesChartData{ x= "Sep", y= 50, y1= 34 },
    new MultipleAxesChartData{ x= "Oct", y= 30, y1= 32 },
    new MultipleAxesChartData{ x= "Nov", y= 35, y1= 32 },
    new MultipleAxesChartData{ x= "Dec", y= 35, y1= 31 }
};
ViewBag.dataSource = chartData;
return View();
}
public class MultipleAxesChartData
{
    public string x;
    public double y;
    public double

```

For spanning the vertical axis along multiple row, you can use [Span](#) property of an axis.

CSHTML

```

@Html.EJS().Chart("container").
Rows(rows =>
{
    rows.Height("50%").Add();
    rows.Height("50%").Add();
}).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("y").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).YAxisName("yAxis").XName("x").Marker(ViewBag.marker).YName("y1").DataSource(ViewBag.dataSource).Name("Japan").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Interval(1).LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90)

```

```

        ).PrimaryYAxis(py => py.span(2))
        .Axes(ax =>
        {

ax.OpposedPosition(true).RowIndex(0).Minimum(24).Maximum(36).Interval(2).Name("yAxis").LabelFormat("{value}°C").Add();
        }
        )
    ).Title("Weather Condition JPN vs DEU").ChartArea(area =>
    area.Border(ViewBag.ChartBorder).Tooltip(tt =>
    tt.Enable(true)).LegendSettings(lg=>lg.Visible(false)).Render()
    )

```

ROW-SPAN.CS

```

public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
    {
        new MultipleAxesChartData{ x= "Jan", y= 15, y1= 33 },
        new MultipleAxesChartData{ x= "Feb", y= 20, y1= 31 },
        new MultipleAxesChartData{ x= "Mar", y= 35, y1= 30 },
        new MultipleAxesChartData{ x= "Apr", y= 40, y1= 28 },
        new MultipleAxesChartData{ x= "May", y= 80, y1= 29 },
        new MultipleAxesChartData{ x= "Jun", y= 70, y1= 30 },
        new MultipleAxesChartData{ x= "Jul", y= 65, y1= 33 },
        new MultipleAxesChartData{ x= "Aug", y= 55, y1= 32 },
        new MultipleAxesChartData{ x= "Sep", y= 50, y1= 34 },
        new MultipleAxesChartData{ x= "Oct", y= 30, y1= 32 },
        new MultipleAxesChartData{ x= "Nov", y= 35, y1= 32 },
        new MultipleAxesChartData{ x= "Dec", y= 35, y1= 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class MultipleAxesChartData
{
    public string x;
    public double y;
    public double
}

```

Columns

To split the chart area horizontally into number of columns, use [Columns](#) property of the chart.

- You can allocate space for each column by using the [Width](#) property. The given width can be either in percentage or in pixel.
- To associate a horizontal axis to a particular column, specify its index to [ColumnIndex](#) property of the axis.
- To customize each column's bottom line, use [Border](#) property.

CSHTML

```
@Html.EJS().Chart("container").Columns(columns =>
```

```

    {
        columns.width("50%").Add();
        columns.width("50%").Add();
    })
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("y").DataSource(ViewBag.dataSource).Name("Germany").Add();

        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).YAxisName("yAxis").XName("x").Marker(ViewBag.marker).YName("y1").DataSource(ViewBag.dataSource).Name("Japan").Add();
    }).PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .Interval(1).LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90)
    ).Axes(ax =>
    {
        ax.OpposedPosition(true).RowIndex(0).Minimum(24).Maximum(36).Interval(2).Name("yAxis").LabelFormat("{value}°C").Add();
    }
    )
    ).Title("Weather Condition JPN vs DEU").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Tooltip(tt =>
    tt.Enable(true)).LegendSettings(lg=>lg.Visible(false)).Render()

```

COLUMN.CS

```

public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
    {
        new MultipleAxesChartData{ x= "Jan", y= 15, y1= 33 },
        new MultipleAxesChartData{ x= "Feb", y= 20, y1= 31 },
        new MultipleAxesChartData{ x= "Mar", y= 35, y1= 30 },
        new MultipleAxesChartData{ x= "Apr", y= 40, y1= 28 },
        new MultipleAxesChartData{ x= "May", y= 80, y1= 29 },
        new MultipleAxesChartData{ x= "Jun", y= 70, y1= 30 },
        new MultipleAxesChartData{ x= "Jul", y= 65, y1= 33 },
        new MultipleAxesChartData{ x= "Aug", y= 55, y1= 32 },
        new MultipleAxesChartData{ x= "Sep", y= 50, y1= 34 },
        new MultipleAxesChartData{ x= "Oct", y= 30, y1= 32 },
        new MultipleAxesChartData{ x= "Nov", y= 35, y1= 32 },
        new MultipleAxesChartData{ x= "Dec", y= 35, y1= 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class MultipleAxesChartData
{
    public string x;
    public double y;
    public double

```


For spanning the horizontal axis along multiple column, you can use [Span](#) property of an axis.

CSHTML

```
@Html.EJS().Chart("container").Columns(columns =>
{
    columns.width("50%").Add();
    columns.width("50%").Add();
})
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("y").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).YAxisName("yAxis").XName("x").Marker(ViewBag.marker).YName("y1").DataSource(ViewBag.dataSource).Name("Japan").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.Interval(1).LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90).Span(2)
).Axes(ax =>
{
    ax.OpposedPosition(true).RowIndex(0).Minimum(24).Maximum(36).Interval(2).Name("yAxis").LabelFormat("{value}°C").Add();
}
)
).Title("Weather Condition JPN vs DEU").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Tooltip(tt =>
tt.Enable(true)).LegendSettings(lg=>lg.Visible(false)).Render()
```

COLUMN-SPAN.CS

```
public ActionResult Index()
{
    List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
{
    new MultipleAxesChartData{ x= "Jan", y= 15, y1= 33 },
    new MultipleAxesChartData{ x= "Feb", y= 20, y1= 31 },
    new MultipleAxesChartData{ x= "Mar", y= 35, y1= 30 },
    new MultipleAxesChartData{ x= "Apr", y= 40, y1= 28 },
    new MultipleAxesChartData{ x= "May", y= 80, y1= 29 },
    new MultipleAxesChartData{ x= "Jun", y= 70, y1= 30 },
    new MultipleAxesChartData{ x= "Jul", y= 65, y1= 33 },
    new MultipleAxesChartData{ x= "Aug", y= 55, y1= 32 },
    new MultipleAxesChartData{ x= "Sep", y= 50, y1= 34 },
    new MultipleAxesChartData{ x= "Oct", y= 30, y1= 32 },
    new MultipleAxesChartData{ x= "Nov", y= 35, y1= 32 },
    new MultipleAxesChartData{ x= "Dec", y= 35, y1= 31 }
};
    ViewBag.dataSource = chartData;
    return View();
}
```

```

    }
    public class MultipleAxesChartData
    {
        public string x;
        public double y;
        public double

```

Chart Types

Line Chart in ASP.NET MVC Charts Component

Line

To render a line series, use series [Type](#) as [Line](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

LINE.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class AxisLabelData
{
    public string x;

```

```

        public double y;
    }

```

Multicolored line

To render a multicolored line series, use the series type as [MultiColoredLine](#). Here, the individual colors to the data can be mapped by using [PointColorMapping](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.MultiColoredLine).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    PointColorMapping("color").
    Name("Gold").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

MULTI-LINE.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4, color="red" },
        new AxisLabelData { x= "India", y= 61.3, color="green" },
        new AxisLabelData { x= "Pakistan", y= 20.4, color="#ff0097" },
        new AxisLabelData { x= "Germany", y= 65.1, color="crimson" },
        new AxisLabelData { x= "Australia", y= 15.8, color="blue" },
        new AxisLabelData { x= "Italy", y= 29.2, color="darkorange" },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
    public string color;
}

```

Series customization

The following properties can be used to customize the [Line](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [Width](#) – Specifies the width for series.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Width("3").
    Opacity("0.5").
    DashArray("5, 5").
    Fill("blue").
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medals").Render()
```

SERIES-LINE.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

See Also

- [Data Label](#)
- [Tooltip](#)

Step line chart in ASP.NET MVC Charts component

Step line

To render a step line series, use series [Type](#) as [StepLine](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StepLine).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

STEPLINE.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

Series customization

The following properties can be used to customize the [Step Line](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [Width](#) – Specifies the width for series.
- [DashArray](#) – Specifies the dashes for series.
- [Step](#) – Specifies the position of the step for the series.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StepLine).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Width(3).
    Opacity(0.5).
    DashArray("5,5").
    Fill("blue").
    Step(Syncfusion.EJ2.Charts.StepPosition.Left).
    Add();
}))
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medals")
.Render())
```

STEPLINESERIES.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

```
}
```

See also

- [Data Label](#)
- [Tooltip](#)

Stacked Line Chart in ASP.NET MVC Charts Component

Stacked Line

To render a stacked line series, use series [Type](#) as [StackingLine](#).

CSHTML

```
@Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y").DataSource(ViewBag.dataSource)
    .Name("John").DashArray("5,1").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y1").DataSource(ViewBag.dataSource)
    .Name("Peter").DashArray("5,1").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y2").DataSource(ViewBag.dataSource)
    .Name("Steve").DashArray("5,1").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y3").DataSource(ViewBag.dataSource)
    .Name("Charle").DashArray("5,1").Add();
})
.PrimaryXAxis(px => px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Family Expense for Month")
.Render()
```

STACKED-LINE.CS

```
public ActionResult Index()
{
    List<StackedLineChartData> chartData = new
List<StackedLineChartData>
{
    new StackedLineChartData { x= "Food", y= 90, y1= 40, y2= 70,
y3= 120 },
    new StackedLineChartData { x= "Transport", y= 80, y1= 90,
y2= 110, y3= 70 },
    new StackedLineChartData { x= "Medical", y= 50, y1= 80, y2=
120, y3= 50 },
    new StackedLineChartData { x= "Clothes", y= 70, y1= 30, y2=
60, y3= 180 },
    new StackedLineChartData { x= "Personal Care", y= 30, y1=
80, y2= 80, y3= 30 },
}
```

```

        new StackedLineChartData { x= "Books", y= 10, y1= 40, y2=
30, y3= 270 },
        new StackedLineChartData { x= "Fitness", y= 100, y1= 30, y2=
70, y3= 40 },
        new StackedLineChartData { x= "Electricity", y= 55, y1= 95,
y2= 55, y3= 75 },
        new StackedLineChartData { x= "Tax", y= 20, y1= 50, y2= 40,
y3= 65 },
        new StackedLineChartData { x= "Pet Care", y= 40, y1= 20, y2=
80, y3= 95 },
        new StackedLineChartData { x= "Education", y= 45, y1= 15,
y2= 45, y3= 195 },
        new StackedLineChartData { x= "Entertainment", y= 75, y1=
45, y2= 65, y3= 115 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class StackedLineChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Series customization

The following properties can be used to customize the [Stacked Line](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [Width](#) – Specifies the width for series.
- [DashArray](#) – Specifies the dashes for series.

CSHTML

```

@Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y").DataSource(ViewBag.dataSource)
    .Name("John").DashArray("5,1").Width(2).Opacity(0.7).Fill("blue").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y1").DataSource(ViewBag.dataSource)
    .Name("Peter").DashArray("5,1").Width(2).Opacity(0.7).Fill("green").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y2").DataSource(ViewBag.dataSource)
    .Name("Steve").DashArray("5,1").Width(2).Opacity(0.7).Fill("red").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y3").DataSource(ViewBag.dataSource)

```



```

.Name("Charle").DashArray("5,1").Width(2).Opacity(0.7).Fill("black").Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .Interval(1)
)
.PrimaryYAxis(py =>
    py.Title("Expense")
    .Interval(100)
    .LabelFormat("${value}")
)
.Title("Family Expense for Month")
.Render()

```

STACKED-LINE-SERIES.CS

```

public ActionResult Index()
{
    List<StackedLineChartData> chartData = new
List<StackedLineChartData>
    {
        new StackedLineChartData { x= "Food", y= 90, y1= 40, y2= 70,
y3= 120 },
        new StackedLineChartData { x= "Transport", y= 80, y1= 90,
y2= 110, y3= 70 },
        new StackedLineChartData { x= "Medical", y= 50, y1= 80, y2=
120, y3= 50 },
        new StackedLineChartData { x= "Clothes", y= 70, y1= 30, y2=
60, y3= 180 },
        new StackedLineChartData { x= "Personal Care", y= 30, y1=
80, y2= 80, y3= 30 },
        new StackedLineChartData { x= "Books", y= 10, y1= 40, y2=
30, y3= 270 },
        new StackedLineChartData { x= "Fitness", y= 100, y1= 30, y2=
70, y3= 40 },
        new StackedLineChartData { x= "Electricity", y= 55, y1= 95,
y2= 55, y3= 75 },
        new StackedLineChartData { x= "Tax", y= 20, y1= 50, y2= 40,
y3= 65 },
        new StackedLineChartData { x= "Pet Care", y= 40, y1= 20, y2=
80, y3= 95 },
        new StackedLineChartData { x= "Education", y= 45, y1= 15,
y2= 45, y3= 195 },
        new StackedLineChartData { x= "Entertainment", y= 75, y1=
45, y2= 65, y3= 115 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class StackedLineChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;

```

```
        public double y3;
    }
```

See Also

- [Data Label](#)
- [Tooltip](#)

100% Stacked Line in ASP.NET MVC Charts Component

100% Stacked Line

To render a 100% stacked line series, use series [Type](#) as [StackingLine100](#).

CSHTML

```
@Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine100)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y").DataSource(ViewBag.dataSource)
    .Name("John").DashArray("5,1").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y1").DataSource(ViewBag.dataSource)
    .Name("Peter").DashArray("5,1").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y2").DataSource(ViewBag.dataSource)
    .Name("Steve").DashArray("5,1").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y3").DataSource(ViewBag.dataSource)
    .Name("Charle").DashArray("5,1").Add();
})
.PrimaryXAxis(px => px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()
```

STACKED-LINE-100.CS

```
public ActionResult Index()
{
    List<StackedLineChartData100> chartData = new
List<StackedLineChartData100>
    {
        new StackedLineChartData100 { x= "Food", y= 90, y1= 40, y2=
70, y3= 120 },
        new StackedLineChartData100 { x= "Transport", y= 80, y1= 90,
y2= 110, y3= 70 },
        new StackedLineChartData100 { x= "Medical", y= 50, y1= 80,
y2= 120, y3= 50 },
        new StackedLineChartData100 { x= "Clothes", y= 70, y1= 30,
y2= 60, y3= 180 },
        new StackedLineChartData100 { x= "Personal Care", y= 30, y1=
80, y2= 80, y3= 30 },
    }
```

```

30, y3= 270 },
        new StackedLineChartData100 { x= "Books", y= 10, y1= 40, y2=
        new StackedLineChartData100 { x= "Fitness", y= 100, y1= 30,
y2= 70, y3= 40 },
        new StackedLineChartData100 { x= "Electricity", y= 55, y1=
95, y2= 55, y3= 75 },
        new StackedLineChartData100 { x= "Tax", y= 20, y1= 50, y2=
40, y3= 65 },
        new StackedLineChartData100 { x= "Pet Care", y= 40, y1= 20,
y2= 80, y3= 95 },
        new StackedLineChartData100 { x= "Education", y= 45, y1= 15,
y2= 45, y3= 195 },
        new StackedLineChartData100 { x= "Entertainment", y= 75, y1=
45, y2= 65, y3= 115 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class StackedLineChartData100
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Series customization

The following properties can be used to customize the [100% Stacked Line](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [Width](#) – Specifies the width for series.
- [DashArray](#) – Specifies the dashes for series.

CSHTML

```

@Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine100)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y").DataSource(ViewBag.dataSource)
    .Name("John").DashArray("5,1").Width(2).Opacity(0.7).Fill("blue").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine100)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y1").DataSource(ViewBag.dataSource)
    .Name("Peter").DashArray("5,1").Width(2).Opacity(0.7).Fill("green").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine100)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y2").DataSource(ViewBag.dataSource)
    .Name("Steve").DashArray("5,1").Width(2).Opacity(0.7).Fill("red").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingLine100)
    .Marker(mkr =>
mkr.Visible(true)).XName("x").YName("y3").DataSource(ViewBag.dataSource)

```

```
.Name("Charle").DashArray("5,1").Width(2).Opacity(0.7).Fill("black").Add();
})
.PrimaryXAxis(px => px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Family Expense for Month")
.Render()
```

STACKED-LINE-100-SERIES.CS

```
public ActionResult Index()
{
    List<StackedLineChartData> chartData = new
List<StackedLineChartData>
{
    new StackedLineChartData { x= "Food", y= 90, y1= 40, y2= 70,
y3= 120 },
    new StackedLineChartData { x= "Transport", y= 80, y1= 90,
y2= 110, y3= 70 },
    new StackedLineChartData { x= "Medical", y= 50, y1= 80, y2=
120, y3= 50 },
    new StackedLineChartData { x= "Clothes", y= 70, y1= 30, y2=
60, y3= 180 },
    new StackedLineChartData { x= "Personal Care", y= 30, y1=
80, y2= 80, y3= 30 },
    new StackedLineChartData { x= "Books", y= 10, y1= 40, y2=
30, y3= 270 },
    new StackedLineChartData { x= "Fitness", y= 100, y1= 30, y2=
70, y3= 40 },
    new StackedLineChartData { x= "Electricity", y= 55, y1= 95,
y2= 55, y3= 75 },
    new StackedLineChartData { x= "Tax", y= 20, y1= 50, y2= 40,
y3= 65 },
    new StackedLineChartData { x= "Pet Care", y= 40, y1= 20, y2=
80, y3= 95 },
    new StackedLineChartData { x= "Education", y= 45, y1= 15,
y2= 45, y3= 195 },
    new StackedLineChartData { x= "Entertainment", y= 75, y1=
45, y2= 65, y3= 115 }
};
    ViewBag.dataSource = chartData;
    return View();
}

public class StackedLineChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}
```

See Also

- [Data Label](#)
- [Tooltip](#)

Spline in ASP.NET MVC Charts Component

Spline

To render a spline series, use series [Type](#) as [Spline](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

SPLINE.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

Type of spline

To specify the type of Spline Chart, use [SplineType](#) property. The spline types are **Clamped**, **Cardinal**, **Monotonic** and **Natural**.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).
    SplineType(Syncfusion.EJ2.Charts.SplineType.Cardinal)
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medals").Width("60%").Render()
```

SPLINETYPE.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

Series customization

The following properties can be used to customize the [Spline](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [Width](#) – Specifies the width for series.
- [DashArray](#) – Specifies the dashes for series.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).
```

```

        XName("x").
        YName("y").
        DashArray("5,5").
        Fill("blue").
        Opacity(0.5).
        Width(3).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Title("Olympic Medals").Width("60%").Render()

```

SPLINE-SERIES.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Area in ASP.NET MVC Charts Component

Area

To render a area series, use series [Type](#) as [Area](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
Name("Gold").
Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

AREA.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4, color="red" },
        new AxisLabelData { x= "India", y= 61.3, color="green" },
        new AxisLabelData { x= "Pakistan", y= 20.4, color="#ff0097" },
        new AxisLabelData { x= "Germany", y= 65.1, color="crimson" },
        new AxisLabelData { x= "Australia", y= 15.8, color="blue" },
        new AxisLabelData { x= "Italy", y= 29.2, color="darkorange" },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
    public string color;
}

```

Multicolored area

To render a multicolored area series, use the series type as [MultiColoredArea](#). The required [Segments](#) of the series can be customized using the [Value](#), [Color](#), and [DashArray](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
XName("x").
YName("y").
PointColorMapping("Color").
DataSource(ViewBag.dataSource).
Name("Gold").

```



```

        Add();
    })
    .PrimaryXAxis(px =>
        px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)
    )
    .Title("Olympic Medal Counts - RIO").Render()

```

MULTICOLOR-AREA.CS

```

public IActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData{ x= 2005, y= 28 },
        new ChartData{ x= 2006, y= 25},
        new ChartData{ x= 2007, y= 26 },
        new ChartData{ x= 2008, y= 27 },
        new ChartData{ x= 2009, y= 32},
        new ChartData{ x= 2010, y= 35 },
        new ChartData{ x= 2011, y= 25 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
}

```

Series customization

The following properties can be used to customize the [Area](#) series.

- [Fill](#) – Specifies the color of the area series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
    XName("x").
    YName("y").
    Opacity(0.3).
    Fill("blue").
    Border(br => br.Width(0)).
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))

```

```
.Render()
```

AREA-SERIES.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

Area Border

The **Width** and **Fill** properties in the [Border](#) can be used to customize the border of all area type series.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Border(Width:2).
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

BORDER.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4, color="red"
    },
        new AxisLabelData { x= "India", y= 61.3, color="green" },
    }
```

```

        new AxisLabelData { x= "Pakistan", y= 20.4, color="#ff0097"
    },
        new AxisLabelData { x= "Germany", y= 65.1, color="crimson"
    },
        new AxisLabelData { x= "Australia", y= 15.8, color="blue" },
        new AxisLabelData { x= "Italy", y= 29.2, color="darkorange"
    }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class AxisLabelData
{
    public string x;
    public double y;
    public string color;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Range Area in ASP.NET MVC Charts Component

Range Area

To render a range area series, use series [Type](#) as [RangeArea](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeArea).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()

```

RANGEAREA.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "Sun", low= 2.5, high= 9.8 },
        new AxisLabelData { x= "Mon", low= 4.7, high= 11.4 },
        new AxisLabelData { x= "Tue", low= 6.4, high= 14.4 },
        new AxisLabelData { x= "Wed", low= 9.6, high= 17.2 },
        new AxisLabelData { x= "Thu", low= 7.5, high= 15.1 },
        new AxisLabelData { x= "Fri", low= 3.0, high= 10.5 },
    }
}

```

```

        new AxisLabelData { x= "Sat", low= 1.2, high= 7.9 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double low;
    public double high;
}

```

Series customization

The following properties can be used to customize the [Range Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeArea).
    XName("x").
    High("high").
    Low("low").
    Border(br => br.Width(2).Color("red")).
    Opacity(0.5).
    DashArray("5,5").
    Fill("blue").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()

```

RANGEAREA-SERIES.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "Sun", low= 2.5, high= 9.8 },
        new AxisLabelData { x= "Mon", low= 4.7, high= 11.4 },
        new AxisLabelData { x= "Tue", low= 6.4, high= 14.4 },
        new AxisLabelData { x= "Wed", low= 9.6, high= 17.2 },
        new AxisLabelData { x= "Thu", low= 7.5, high= 15.1 },
        new AxisLabelData { x= "Fri", low= 3.0, high= 10.5 },
        new AxisLabelData { x= "Sat", low= 1.2, high= 7.9 }
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class AxisLabelData
    {
        public string x;
        public double low;
        public double high;
    }

```

See Also

- [Data Label](#)
- [Tooltip](#)

Range step area in ASP.NET MVC Charts component

Range step area

To render the range step area series, use the series [Type](#) as a [RangeStepArea](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeStepArea).
    Opacity(0.4).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Name("India").
    Add();
})
.PrimaryXAxis(px =>
    px.MajorGridLines(mg => mg.Width(0))

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .PrimaryYAxis(py => py.LabelFormat("{value}°C")
    .Minimum(0)
    .Maximum(40)
    .MajorTickLines(mt => mt.Width(0))
    .LineStyle(ls => ls.Width(0))
)
.Title("Monthly Temperature Range").Render()

```

RANGE-STEP-AREA.CS

```

public IActionResult Index ()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x = "Jan", high = 14, low = 4, high1 = 29,
low1 = 19 },
        new ChartData { x = "Feb", high = 17, low = 7, high1 = 32,
low1 = 22 },
    }
}

```

```

low1 = 25 },
        new ChartData { x = "Mar", high = 20, low = 10, high1 = 35,
low1 = 27 },
        new ChartData { x = "Apr", high = 22, low = 12, high1 = 37,
low1 = 25 },
        new ChartData { x = "May", high = 20, low = 10, high1 = 35,
low1 = 22 },
        new ChartData { x = "Jun", high = 17, low = 7, high1 = 32,
low1 = 20 },
        new ChartData { x = "Jul", high = 15, low = 5, high1 = 30,
low1 = 22 },
        new ChartData { x = "Aug", high = 17, low = 7, high1 = 32,
low1 = 25 },
        new ChartData { x = "Sep", high = 20, low = 10, high1 = 35,
low1 = 27 },
        new ChartData { x = "Oct", high = 22, low = 12, high1 = 37,
low1 = 25 },
        new ChartData { x = "Nov", high = 20, low = 10, high1 = 35,
low1 = 22 }
        new ChartData { x = "Dec", high = 17, low = 7, high1 = 32,
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double high;
    public double low;
    public double high1;
    public double low1;
}

```

Series customization

The following properties can be used to customize the [Range Step Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.
- [Step](#) – Specifies the position of the step for the series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeStepArea).
    Opacity(0.7).
    Fill("blue").
    DashArray("5,5").
    Border(br => br.Width(2).Color("black")).
    XName("x").
    High("high1").
    Low("low1").
}

```

```

        Step(Syncfusion.EJ2.Charts.StepPosition.Center).
        DataSource(ViewBag.dataSource).
        Name("India").
        Add();
    })
    .PrimaryXAxis(px =>
        px.MajorGridLines(mg => mg.Width(0))

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(py =>
        py.LabelFormat("{value} °C")
        .Minimum(0)
        .Maximum(40)
        .MajorTickLines(mt => mt.Width(0))
        .LineStyle(ls => ls.Width(0))
    )
    .Title("Monthly Temperature Range").Render()

```

RANGE-STEP-AREA-SERIES.CS

```

public IActionResult Index ()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x = "Jan", high1 = 29, low1 = 19 },
        new ChartData { x = "Feb", high1 = 32, low1 = 22 },
        new ChartData { x = "Mar", high1 = 35, low1 = 25 },
        new ChartData { x = "Apr", high1 = 37, low1 = 27 },
        new ChartData { x = "May", high1 = 35, low1 = 25 },
        new ChartData { x = "Jun", high1 = 32, low1 = 22 },
        new ChartData { x = "Jul", high1 = 30, low1 = 20 },
        new ChartData { x = "Aug", high1 = 32, low1 = 22 },
        new ChartData { x = "Sep", high1 = 35, low1 = 25 },
        new ChartData { x = "Oct", high1 = 37, low1 = 27 },
        new ChartData { x = "Nov", high1 = 35, low1 = 25 },
        new ChartData { x = "Dec", high1 = 32, low1 = 22 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double high1;
    public double low1;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

Spline Range Area in ASP.NET MVC Charts Component

Spline Range Area

The Spline Range Area Chart is used to display continuous data points as a set of splines that vary between high and low values over intervals of time and across different categories.

To render a spline range area series, use series [Type](#) as [SplineRangeArea](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineRangeArea).
    Opacity(0.4).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Name("England").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineRangeArea).
    Opacity(0.4).
    XName("x").
    High("high1").
    Low("low1").
    DataSource(ViewBag.dataSource).
    Name("India").
    Add();
})
.PrimaryXAxis(px =>
    px.MajorGridLines(mg => mg.Width(0))

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.PrimaryYAxis(py =>
    py.LabelFormat("{value} °C")
    .Minimum(0)
    .Maximum(40)
    .MajorTickLines(mt => mt.Width(0))
    .LineStyle(ls => ls.Width(0))
)
.Render()
```

SPLINERANGEAREA.CS

```
public IActionResult Index () {
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x = "Jan", high = 14, low = 4, high1 = 29, low1 = 19
    },
        new ChartData { x = "Feb", high = 17, low = 7, high1 = 32, low1 = 22
    },
        new ChartData { x = "Mar", high = 20, low = 10, high1 = 35, low1 =
25 },
        new ChartData { x = "Apr", high = 22, low = 12, high1 = 37, low1 =
27 },
    }
```



```

25     new ChartData { x = "May", high = 20, low = 10, high1 = 35, low1 =
    },
    new ChartData { x = "Jun", high = 17, low = 7, high1 = 32, low1 = 22
    },
    new ChartData { x = "Jul", high = 15, low = 5, high1 = 30, low1 = 20
    },
    new ChartData { x = "Aug", high = 17, low = 7, high1 = 32, low1 = 22
    },
25     new ChartData { x = "Sep", high = 20, low = 10, high1 = 35, low1 =
27     },
    new ChartData { x = "Oct", high = 22, low = 12, high1 = 37, low1 =
25     },
    new ChartData { x = "Nov", high = 20, low = 10, high1 = 35, low1 =
    },
    new ChartData { x = "Dec", high = 17, low = 7, high1 = 32, low1 = 22
    }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double high;
    public double low;
    public double high1;
    public double low1;
}

```

Series customization

The following properties can be used to customize the [Spline Range Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineRangeArea).
    Opacity(0.5).
    Border(br =>
br.Width(2).Color("red").DashArray("5,5").Fill("blue").
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Name("England").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineRangeArea).
    Opacity(0.5).
    Border(br =>
br.Width(2).Color("yellow").DashArray("5,5").Fill("green").

```

```

        XName("x").
        High("high1").
        Low("low1").
        DataSource(ViewBag.dataSource).
        Name("India").
        Add();
    })
    .PrimaryXAxis(px =>
        px.MajorGridLines(mg => mg.Width(0))

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(py =>
        py.LabelFormat("{value}°C")
        .Minimum(0)
        .Maximum(40)
        .MajorTickLines(mt => mt.Width(0))
        .LineStyle(ls => ls.Width(0))
    )
    .Render()

```

SPLINERANGEAREA-SERIES.CS

```

public IActionResult Index () {
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x = "Jan", high = 14, low = 4, high1 = 29, low1 = 19
    },
        new ChartData { x = "Feb", high = 17, low = 7, high1 = 32, low1 = 22
    },
        new ChartData { x = "Mar", high = 20, low = 10, high1 = 35, low1 =
25 },
        new ChartData { x = "Apr", high = 22, low = 12, high1 = 37, low1 =
27 },
        new ChartData { x = "May", high = 20, low = 10, high1 = 35, low1 =
25 },
        new ChartData { x = "Jun", high = 17, low = 7, high1 = 32, low1 = 22
    },
        new ChartData { x = "Jul", high = 15, low = 5, high1 = 30, low1 = 20
    },
        new ChartData { x = "Aug", high = 17, low = 7, high1 = 32, low1 = 22
    },
        new ChartData { x = "Sep", high = 20, low = 10, high1 = 35, low1 =
25 },
        new ChartData { x = "Oct", high = 22, low = 12, high1 = 37, low1 =
27 },
        new ChartData { x = "Nov", high = 20, low = 10, high1 = 35, low1 =
25 },
        new ChartData { x = "Dec", high = 17, low = 7, high1 = 32, low1 = 22
    }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData

```

```
{
    public string x;
    public double high;
    public double low;
    public double high1;
    public double low1;
}
```

See Also

- [Data Label](#)
- [Tooltip](#)

Stacked Area in ASP.NET MVC Charts Component

Stacked Area

To render a stacked area series, use series [Type](#) as [StackingArea](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    Name("Platinum").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

STACKEDAREA.CS

```
public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
    }
```

```

        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Series customization

The following properties can be used to customize the [Stacked Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Border(br => br.Width(2).Color("black")).
    Fill("pink").
    Opacity(0.7).
    DashArray("5,5").
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    Border(br => br.Width(2).Color("black")).
    Fill("blue").
    Opacity(0.7).
    DashArray("5,5").
    Name("Silver").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea).
    XName("x").

```

```

        YName("y2").
        DataSource(ViewBag.dataSource).
        Border(br => br.Width(2).Color("black")).
        Fill("green").
        Opacity(0.7).
        DashArray("5,5").
        Name("Silver").
        Add();
    })
    .PrimaryXAxis(px =>
        px.Interval(1)
            .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
            .IsIndexed(true)
    )
    .Title("Olympic Medal Counts - RIO").Render()

```

STACKEDAREA-SERIES.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

100% Stacked Area in ASP.NET MVC Charts Component

100% Stacked Area Chart

To render a 100% stacked area series, use series [Type](#) as [StackingArea100](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea100).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea100).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea100).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    Name("Platinum").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

STACKEDAREA100.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Series customization

The following properties can be used to customize the [100% Stacked Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea100).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Fill("pink").
    Opacity(0.7).
    DashArray("5,5").
    Border(br => br.Width(2).Color("black")).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea100).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Fill("blue").
    Opacity(0.7).
    DashArray("5,5").
    Border(br => br.Width(2).Color("black")).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingArea100).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    Name("Platinum").
    Fill("green").
    Opacity(0.7).
    DashArray("5,5").
    Border(br => br.Width(2).Color("black")).
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

STACKEDAREA100-SERIES.CS

```
public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
```

```

        {
            new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
            new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
            new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
            new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
            new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
            new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
            new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
            new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
            new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
            new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class PolarData
    {
        public double x;
        public double y;
        public double y1;
        public double y2;
    }
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Stacked step area in ASP.NET MVC Charts component

Stacked step area

To render a stacked step area series, use series [Type](#) as [StackingStepArea](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingStepArea).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingStepArea).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingStepArea).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    Add();
})
.Render()

```


STACKED-STEP-AREA.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Series customization

The following properties can be used to customize the [Stacked Step Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.
- [Step](#) – Specifies the position of the step for the series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingStepArea).
    XName("x").
    YName("y").
    Border(br => br.Width(2).Color("black")).
    Fill("pink").
    Opacity(0.7).
    DashArray("5,5").
    Step(Syncfusion.EJ2.Charts.StepPosition.Center).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingStepArea).
    XName("x").
    YName("y1").
    Border(br => br.Width(2).Color("black")).

```

```

        Fill("blue").
        Opacity(0.7).
        DashArray("5,5").
        Step(Syncfusion.EJ2.Charts.StepPosition.Center).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingStepArea).
        XName("x").
        YName("y2").
        Border(br => br.Width(2).Color("black")).
        Fill("green").
        Opacity(0.7).
        DashArray("5,5").
        Step(Syncfusion.EJ2.Charts.StepPosition.Center).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .Render()

```

STACKEDSTEPAREA-SERIES.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1,      y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

Step area in ASP.NET MVC Charts component

Step area

To render a step area series, use series [Type](#) as [StepArea](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StepArea).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()
```

STEPAREA.CS

```
public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData { x= "Sun", low= 2.5, high= 9.8 },
        new PolarData { x= "Mon", low= 4.7, high= 11.4 },
        new PolarData { x= "Tue", low= 6.4, high= 14.4 },
        new PolarData { x= "Wed", low= 9.6, high= 17.2 },
        new PolarData { x= "Thu", low= 7.5, high= 15.1 },
        new PolarData { x= "Fri", low= 3.0, high= 10.5 },
        new PolarData { x= "Sat", low= 1.2, high= 7.9 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double low;
    public double high;
}
```

Series customization

The following properties can be used to customize the [Step Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.
- [Step](#) – Specifies the position of the step for the series.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StepArea).
    XName("x").
    YName("high").
    Fill("blue").
    Border(br => br.Width(2).Color("black")).
    Opacity(0.5).
    DashArray("5,5").
    Step(Syncfusion.EJ2.Charts.StepPosition.Right).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StepArea).
    XName("x").
    YName("low").
    Fill("green").
    Border(br => br.Width(2).Color("black")).
    Opacity(0.5).
    DashArray("5,5").
    Step(Syncfusion.EJ2.Charts.StepPosition.Right).
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render())
```

STEPAREA-SERIES.CS

```
public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData { x= "Sun", low= 2.5, high= 9.8 },
        new PolarData { x= "Mon", low= 4.7, high= 11.4 },
        new PolarData { x= "Tue", low= 6.4, high= 14.4 },
        new PolarData { x= "Wed", low= 9.6, high= 17.2 },
        new PolarData { x= "Thu", low= 7.5, high= 15.1 },
        new PolarData { x= "Fri", low= 3.0, high= 10.5 },
        new PolarData { x= "Sat", low= 1.2, high= 7.9 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public string x;
    public double low;
    public double high;
}
```

See also

- [Data Label](#)
- [Tooltip](#)

Spline Area in ASP.NET MVC Charts Component

Spline Area

To render a spline series, use series [Type](#) as [Spline](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineArea).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medals").Render()
```

SPLINEAREA.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

Series customization

The following properties can be used to customize the [Spline Area](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineArea).
    XName("x").
    YName("y").
    Opacity(0.5).
    DashArray("5,5").
    Fill("blue").
    DataSource(ViewBag.dataSource).
    Border(br => br.Width(2).Color("black")).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medals").Render()
```

SPLINEAREA-SERIES.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

See Also

- [Data Label](#)
- [Tooltip](#)

Column Chart in ASP.NET MVC Charts Component

Column

To render a column series, use series [Type](#) as [Column](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

COLUMN.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

Column space and width

The [ColumnSpacing](#) and [ColumnWidth](#) properties are used to customize the space between columns.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("y").
```

```

        DataSource(ViewBag.dataSource).
        ColumnSpacing(0.2).
        ColumnWidth(0.2).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Render()

```

COLUMN-SPACE.CS

```

public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}

```

Grouped column

You can use the [GroupName](#) property to group the data points in the column type charts. Data points with same group name are grouped together.

CSHTML

```

@Html.EJS().Chart("container")
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .MajorGridLines(mg => mg.Width(0))
        .Interval(1))
    .PrimaryYAxis(py =>
        py.LabelStyle(ls => ls.Color("transparent"))
        .LineStyle(ls => ls.Width(0))
        .MajorTickLines(mt => mt.Width(0))
        .MajorGridLines(mg => mg.Width(0))
        .MinorGridLines(mg => mg.Width(0)))
    .ChartArea(area => area.Border(br => br.Width(0)))

```



```

        .Series(series =>
        {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("Year").YName("USA_Total")
            .Width(2).ColumnWidth(0.7).ColumnSpacing(0.1).Name("USA Total").GroupName("USA")
            .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
            .Font(font =>
font.FontWeight("600").Color("#FFFFFF")))).DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("Year").YName("USA_Gold")
            .Width(2).ColumnWidth(0.5).Opacity(0.4).ColumnSpacing(0.1).Name("USA Gold").GroupName("USA")
            .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
            .Font(font =>
font.FontWeight("600").Color("#FFFFFF")))).DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("Year").YName("UK_Total")
            .Width(2).ColumnWidth(0.7).ColumnSpacing(0.1).Name("UK Total").GroupName("UK")
            .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
            .Font(font =>
font.FontWeight("600").Color("#FFFFFF")))).DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("Year").YName("UK_Gold")
            .Width(2).ColumnWidth(0.5).ColumnSpacing(0.1).Name("UK Gold").GroupName("UK")
            .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
            .Font(font =>
font.FontWeight("600").Color("#FFFFFF")))).DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("Year").YName("China_Total")
            .Width(2).ColumnWidth(0.7).ColumnSpacing(0.1).Name("China Total").GroupName("China")
            .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
            .Font(font =>
font.FontWeight("600").Color("#FFFFFF")))).DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("Year").YName("China_Gold")
            .Width(2).ColumnWidth(0.5).ColumnSpacing(0.1).Name("China Gold").GroupName("China")
            .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
            .Font(font =>
font.FontWeight("600").Color("#FFFFFF")))).DataSource(ViewBag.data).Add();

```

```

    })
    .Title("Olympics Medal Tally").Load("load")
    .Tooltip(tooltip => tooltip.Enable(true)).Render()

```

GROUP-COLUMN.CS

```

public ActionResult Index()
{
    List<ColumnData> chartData = new List<ColumnData>
    {
        new ColumnData { Year = "2012", USA_Total = 104, USA_Gold =
46, UK_Total = 65, UK_Gold = 29, China_Total = 91, China_Gold = 38},
        new ColumnData { Year = "2016", USA_Total = 121, USA_Gold =
46, UK_Total = 67, UK_Gold = 27, China_Total = 70, China_Gold = 26},
        new ColumnData { Year = "2020", USA_Total = 113, USA_Gold =
39, UK_Total = 65, UK_Gold = 22, China_Total = 88, China_Gold = 38},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnData
{
    public string Year;
    public double USA_Total;
    public double USA_Gold;
    public double UK_Total;
    public double UK_Gold;
    public double China_Total;
    public double China_Gold;
}

```

Cylindrical column chart

To render a cylindrical column chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
XName("country").
YName("gold").
ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
DataSource(ViewBag.dataSource).
TooltipMappingName("tooltipMappingName").
Add();
}))
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.PrimaryYAxis(py =>
    py.Title("Medal Count")
    .Interval(10).Minimum(0).Maximum(80)
)
.Title("Olympic Gold Medal Counts - RIO")

```

```
.Tooltip(tt =>
tt.Enable(true).Header("<b>${point.tooltip}</b>").Format("Gold Medal:
<b>${point.y}</b>").Render())
```

COLUMN-CYLINDER.CS

```
public ActionResult Index()
{
    List<CylindricalChartData> chartData = new List<CylindricalChartData>
    {
        new CylindricalChartData { country= "USA",          gold= 50,
tooltipMappingName= "USA"      },
        new CylindricalChartData { country= "Japan",        gold= 70,
tooltipMappingName= "Japan"    },
        new CylindricalChartData { country= "Australia",    gold= 60,
tooltipMappingName= "Australia"},
        new CylindricalChartData { country= "France",       gold= 50,
tooltipMappingName= "France"   },
        new CylindricalChartData { country= "Italy",        gold= 40,
tooltipMappingName= "Italy"    },
        new CylindricalChartData { country= "Sweden",       gold= 55,
tooltipMappingName= "Sweden"   }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class CylindricalChartData
{
    public string country;
    public double gold;
    public string tooltipMappingName;
}
```

Series customization

The following properties can be used to customize the [Column](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
Opacity(0.5).
DashArray("5,5").
Fill("blue").
Add();
})
```

```
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.Render()
```

COLUMN-SERIES.CS

```
public ActionResult Index()
{
    List<AxisLabelData> chartData = new List<AxisLabelData>
    {
        new AxisLabelData { x= "South Korea", y= 39.4 },
        new AxisLabelData { x= "India", y= 61.3 },
        new AxisLabelData { x= "Pakistan", y= 20.4 },
        new AxisLabelData { x= "Germany", y= 65.1 },
        new AxisLabelData { x= "Australia", y= 15.8 },
        new AxisLabelData { x= "Italy", y= 29.2 },
        new AxisLabelData { x= "United Kingdom", y= 44.6 },
        new AxisLabelData { x= "Saudi Arabia", y= 9.7 },
        new AxisLabelData { x= "Russia", y= 40.8 },
        new AxisLabelData { x= "Mexico", y= 31 },
        new AxisLabelData { x= "Brazil", y= 75.9 },
        new AxisLabelData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class AxisLabelData
{
    public string x;
    public double y;
}
```

See also

- [Data Label](#)
- [Tooltip](#)

Range Column in ASP.NET MVC Charts Component

Range Column

To render a range column series, use series [Type](#) as [RangeColumn](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeColumn).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).
```

```

        Add();
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeColumn).
        XName("x").
        High("high").
        Low("low").
        DataSource(ViewBag.dataSource1).
        Name("Silver").
        Width(2).
        Add();
    })
    .PrimaryXAxis(px =>
        px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medal Counts - RIO").Render()

```

RANGECOLUMN.CS

```

public ActionResult Index()
{
    List<RangeColumnChartData> chartData = new
List<RangeColumnChartData>
    {
        new RangeColumnChartData { x= "Sun", low= 3.1, high= 10.8 },
        new RangeColumnChartData { x= "Mon", low= 5.7, high= 14.4 },
        new RangeColumnChartData { x= "Tue", low= 8.4, high= 16.9 },
        new RangeColumnChartData { x= "Wed", low= 10.6, high= 19.2 },
        new RangeColumnChartData { x= "Thu", low= 8.5, high= 16.1 },
        new RangeColumnChartData { x= "Fri", low= 6.0, high= 12.5 },
        new RangeColumnChartData { x= "Sat", low= 1.5, high= 6.9 }
    };
    ViewBag.dataSource = chartData;
    List<RangeColumnChartData> chartData1 = new
List<RangeColumnChartData>
    {
        new RangeColumnChartData { x= "Sun", low= 2.5, high= 9.8 },
        new RangeColumnChartData { x= "Mon", low= 4.7, high= 11.4 },
        new RangeColumnChartData { x= "Tue", low= 6.4, high= 14.4 },
        new RangeColumnChartData { x= "Wed", low= 9.6, high= 17.2 },
        new RangeColumnChartData { x= "Thu", low= 7.5, high= 15.1 },
        new RangeColumnChartData { x= "Fri", low= 3.0, high= 10.5 },
        new RangeColumnChartData { x= "Sat", low= 1.2, high= 7.9 }
    };
    ViewBag.dataSource1 = chartData1;
    return View();
}

public class RangeColumnChartData
{
    public string x;
    public double low;
    public double high;
}

```

Series customization

The following properties can be used to customize the [Range Column](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.
- [ColumnSpacing](#) – Specifies the space between the series segments.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeColumn).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Width(2).
    Fill("green").
    ColumnSpacing(0.4).
    DashArray("5,5").
    Opacity(0.7).
    Border(br => br.Width(2).Color("green")).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.RangeColumn).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource1).
    Width(2).
    Fill("red").
    ColumnSpacing(0.4).
    DashArray("5,5").
    Opacity(0.7).
    Border(br => br.Width(2).Color("red")).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()
```

RANGECOLUMN-SERIES.CS

```
public ActionResult Index()
{
    List<RangeColumnChartData> chartData = new
List<RangeColumnChartData>
{
    new RangeColumnChartData { x= "Sun", low= 3.1, high= 10.8 },
    new RangeColumnChartData { x= "Mon", low= 5.7, high= 14.4 },
    new RangeColumnChartData { x= "Tue", low= 8.4, high= 16.9 },
    new RangeColumnChartData { x= "Wed", low= 10.6, high= 19.2 },
    new RangeColumnChartData { x= "Thu", low= 8.5, high= 16.1 },
    new RangeColumnChartData { x= "Fri", low= 6.0, high= 12.5 },
    new RangeColumnChartData { x= "Sat", low= 1.5, high= 6.9 }
};
    ViewBag.dataSource = chartData;
```

```

        List<RangeColumnChartData> chartData1 = new
List<RangeColumnChartData>
    {
        new RangeColumnChartData { x= "Sun", low= 2.5, high= 9.8 },
        new RangeColumnChartData { x= "Mon", low= 4.7, high= 11.4 },
        new RangeColumnChartData { x= "Tue", low= 6.4, high= 14.4 },
        new RangeColumnChartData { x= "Wed", low= 9.6, high= 17.2 },
        new RangeColumnChartData { x= "Thu", low= 7.5, high= 15.1 },
        new RangeColumnChartData { x= "Fri", low= 3.0, high= 10.5 },
        new RangeColumnChartData { x= "Sat", low= 1.2, high= 7.9 }
    };
    ViewBag.dataSource1 = chartData1;
    return View();
}
public class RangeColumnChartData
{
    public string x;
    public double low;
    public double high;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Stacked Column in ASP.NET MVC Charts Component

Stacked column

To render a stacked column series, use series [Type](#) as [StackingColumn](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

STACKEDCOLUMN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Stacking group

You can use the [StackingGroup](#) property to group the stacked columns and 100% stacked columns. Columns with same group name are stacked on top of each other.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    StackingGroup("first").
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    StackingGroup("second").
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```


GROUP.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Cylindrical stacked column chart

To render a cylindrical stacked column chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("y").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Name("UK").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("y1").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Name("Germany").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("y2").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Name("France").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").

```

```

        YName("y3").
        ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
        DataSource(ViewBag.dataSource).
        Name("Italy").
        Add();
    })
    .PrimaryXAxis(px =>
        px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .Title("Years")
    )
    .PrimaryYAxis(py =>
        py.Title("Sales in Billions")
        .Interval(100).Minimum(0).Maximum(700).LabelFormat("{value}B")
    )
    .Render()

```

STACKEDCOLUMN-CYLINDER.CS

```

public ActionResult Index()
{
    List<CylindricalChartData> chartData = new List<CylindricalChartData>
    {
        new CylindricalChartData { x= "2014", y= 111.1, y1= 76.9, y2= 66.1,
y3= 34.1 },
        new CylindricalChartData { x= "2015", y= 127.3, y1= 99.5, y2= 79.3,
y3= 38.2 },
        new CylindricalChartData { x= "2016", y= 143.4, y1= 121.7, y2= 91.3,
y3= 44.0 },
        new CylindricalChartData { x= "2017", y= 159.9, y1= 142.5, y2=
102.4, y3= 51.6 },
        new CylindricalChartData { x= "2018", y= 175.4, y1= 166.7, y2=
112.9, y3= 61.9 },
        new CylindricalChartData { x= "2019", y= 189.0, y1= 182.9, y2=
122.4, y3= 71.5 },
        new CylindricalChartData { x= "2020", y= 202.7, y1= 197.3, y2=
120.9, y3= 82.0 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class CylindricalChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Series customization

The following properties can be used to customize the [Stacked Column](#) series.

- [Fill](#) – Specifies the color of the series.

- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    DashArray("5,5").
    Fill("blue").
    Opacity(0.7).
    Border(br => br.Width(2).Color("black")).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    DashArray("5,5").
    Fill("green").
    Opacity(0.7).
    Border(br => br.Width(2).Color("black")).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()
```

STACKEDCOLUMN-SERIES.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}
```

See also

- [Data Label](#)
- [Tooltip](#)

100% Stacked Column in ASP.NET MVC Charts Component

100% Stacked column

To render a 100% stacked column series, use series [Type](#) as [StackingColumn100](#).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()
```

STACKEDCOLUMN100.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{

```

```

        public string x;
        public double yValue;
        public double yValue1;
    }

```

100% Cylindrical stacked column chart

To render a 100% cylindrical stacked column chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```

@ (Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y").
    Name("UK").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y1").
    Name("Germany").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y2").
    Name("France").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y3").
    Name("Italy").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .Title("Years")
        .LabelFormat("y")
    )
.PrimaryYAxis(py =>
    py.Title("GDP (%) Per Annum")
        .LabelFormat("{value}%")
        .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    )
.Title("Gross Domestic Product Growth")
.Render()

```

STACKEDCOLUMN100-CYLINDER.CS

```

public ActionResult Index()
{
    List<CylindricalChartData> chartData = new List<CylindricalChartData>
    {
        new CylindricalChartData { x= new DateTime(2006, 01, 01), y= 900,
y1= 190, y2= 250, y3= 150 },
        new CylindricalChartData { x= new DateTime(2007, 01, 01), y= 544,
y1= 226, y2= 145, y3= 120 },
        new CylindricalChartData { x= new DateTime(2008, 01, 01), y= 880,
y1= 194, y2= 190, y3= 115 },
        new CylindricalChartData { x= new DateTime(2009, 01, 01), y= 675,
y1= 250, y2= 220, y3= 125 },
        new CylindricalChartData { x= new DateTime(2010, 01, 01), y= 765,
y1= 222, y2= 225, y3= 132 },
        new CylindricalChartData { x= new DateTime(2011, 01, 01), y= 679,
y1= 181, y2= 135, y3= 137 },
        new CylindricalChartData { x= new DateTime(2012, 01, 01), y= 770,
y1= 128, y2= 152, y3= 110 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class CylindricalChartData
{
    public DateTime x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Series customization

The following properties can be used to customize the [100% Stacked Column](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("yValue").
    DashArray("5,5").
    Fill("red").
    Opacity(0.7).
    StackingGroup("Asia").
    Border(br => br.Width(2).Color("black")).
    DataSource(ViewBag.dataSource).
    Add();
}

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
XName("x").
YName("yValue1").
DashArray("5,5").
Fill("blue").
Opacity(0.7).
StackingGroup("Asia").
Border(br => br.Width(2).Color("black")).
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()

```

STACKEDCOLUMN-100-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

Bar Charts in ASP.NET MVC Charts Component

Bar

To render a bar series, use series [Type](#) as [Bar](#).

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).

```

```

XName("x").
YName("yValue").
DataSource(ViewBag.dataSource).
Name("Gold").
Add();
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).
XName("x").
YName("yValue1").
DataSource(ViewBag.dataSource).
Name("Silver").
Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

BAR.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Bar space and width

The [ColumnSpacing](#) and [ColumnWidth](#) properties are used to customize the space between bars.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).
XName("x").
YName("yValue").
DataSource(ViewBag.dataSource).
ColumnSpacing(0.2).

```



```

        ColumnWidth(0.7).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Title("Olympic Medals").Render()

```

BAR-SPACE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46 },
        new ChartData { x= "GBR", yValue= 27 },
        new ChartData { x= "CHN", yValue= 26 },
        new ChartData { x= "UK", yValue= 56 },
        new ChartData { x= "AUS", yValue= 12 },
        new ChartData { x= "IND", yValue= 26 },
        new ChartData { x= "DEN", yValue= 26 },
        new ChartData { x= "MEX", yValue= 34 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double yValue;
}

```

Grouped bar

You can use the [GroupName](#) property to group the data points in the bar type charts. Data points with same group name are grouped together.

CSHTML

```

@Html.EJS().Chart("container")
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .MajorGridLines(mg => mg.Width(0)).Interval(1))
    .PrimaryYAxis(py =>
        py.LabelStyle(ls => ls.Color("transparent")).LineStyle(ls =>
ls.Width(0))
        .MajorTickLines(mt => mt.Width(0))
        .MajorGridLines(mg => mg.Width(0))
        .MinorGridLines(mg => mg.Width(0)))
    .ChartArea(area => area.Border(br => br.Width(0)))
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).XName("Year").YName("
USA_Total")
        .Width(2).ColumnWidth(0.7).ColumnSpacing(0.1).Name("USA
Total").GroupName("USA")
    })

```

```

        .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
        .Font(font =>
font.FontWeight("600").Color("#FFFFFF"))) .DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).XName("Year").YName("
USA_Gold")

.Width(2).ColumnWidth(0.5).Opacity(0.4).ColumnSpacing(0.1).Name("USA
Gold").GroupName("USA")
        .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
        .Font(font =>
font.FontWeight("600").Color("#FFFFFF"))) .DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).XName("Year").YName("
UK_Total")
        .Width(2).ColumnWidth(0.7).ColumnSpacing(0.1).Name("UK
Total").GroupName("UK")
        .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
        .Font(font =>
font.FontWeight("600").Color("#FFFFFF"))) .DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).XName("Year").YName("
UK_Gold")
        .Width(2).ColumnWidth(0.5).ColumnSpacing(0.1).Name("UK
Gold").GroupName("UK")
        .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
        .Font(font =>
font.FontWeight("600").Color("#FFFFFF"))) .DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).XName("Year").YName("
China_Total")
        .Width(2).ColumnWidth(0.7).ColumnSpacing(0.1).Name("China
Total").GroupName("China")
        .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
        .Font(font =>
font.FontWeight("600").Color("#FFFFFF"))) .DataSource(ViewBag.data).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).XName("Year").YName("
China_Gold")
        .Width(2).ColumnWidth(0.5).ColumnSpacing(0.1).Name("China
Gold").GroupName("China")
        .Marker(marker => marker.DataLabel(label =>
label.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top)
        .Font(font =>
font.FontWeight("600").Color("#FFFFFF"))) .DataSource(ViewBag.data).Add();
    })
    .Title("Olympics Medal Tally").Load("load")
    .Tooltip(tooltip => tooltip.Enable(true)).Render()

```

GROUP-BAR.CS

```

public ActionResult Index()
{
    List<ColumnData> chartData = new List<ColumnData>
    {
        new ColumnData { Year = "2012", USA_Total = 104, USA_Gold =
46, UK_Total = 65, UK_Gold = 29, China_Total = 91, China_Gold = 38},
        new ColumnData { Year = "2016", USA_Total = 121, USA_Gold =
46, UK_Total = 67, UK_Gold = 27, China_Total = 70, China_Gold = 26},
        new ColumnData { Year = "2020", USA_Total = 113, USA_Gold =
39, UK_Total = 65, UK_Gold = 22, China_Total = 88, China_Gold = 38},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnData
{
    public string Year;
    public double USA_Total;
    public double USA_Gold;
    public double UK_Total;
    public double UK_Gold;
    public double China_Total;
    public double China_Gold;
}

```

Cylindrical bar chart

To render a cylindrical bar chart, set the [ColumnFacet](#) property to `Cylinder` in the chart series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
Name("India").
Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .Minimum(2005)
    .Maximum(2012)
)
.PrimaryYAxis(py => py
    .Title("Percentage")
    .Interval(1)
    .Minimum(3)
    .Maximum(12)
)
.Title("Unemployment rate in percentage").Render())

```

BAR-CYLINDER.CS

```

public ActionResult Index()

```

```

{
    List<CylinderChartData> chartData = new List<CylinderChartData>
    {
        new CylinderChartData { x= 2006, y= 9 },
        new CylinderChartData { x= 2007, y= 7.8 },
        new CylinderChartData { x= 2008, y= 10.5 },
        new CylinderChartData { x= 2009, y= 8.4 },
        new CylinderChartData { x= 2010, y= 6 },
        new CylinderChartData { x= 2011, y= 11 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class CylinderChartData
{
    public double x;
    public double y;
}

```

Series customization

The following properties can be used to customize the [Bar](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bar).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Fill("blue").
    Opacity(0.5).
    DashArray("5,5").
    Border(br => br.Width(2).Color("black")).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medals").Render()

```

BAR-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46 },
        new ChartData { x= "GBR", yValue= 27 },
        new ChartData { x= "CHN", yValue= 26 },
    }
}

```

```

        new ChartData { x= "UK", yValue= 56 },
        new ChartData { x= "AUS", yValue= 12 },
        new ChartData { x= "IND", yValue= 26 },
        new ChartData { x= "DEN", yValue= 26 },
        new ChartData { x= "MEX", yValue= 34 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double yValue;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

Stacked Bar in ASP.NET MVC Charts Component

Stacked bar

To render a stacked bar series, use series [Type](#) as [StackingBar](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

STACKEDBAR.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {

```

```

        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Stacking group

You can use the [StackingGroup](#) property to group the stacked bar and 100% stacked bar. Columns with same group name are stacked on top of each other.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("yValue").
    StakingGroup("first").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("yValue1").
    StakingGroup("second").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

GROUP.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {

```

```

        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Cylindrical stacked bar chart

To render a cylindrical stacked bar chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
}))
.Render()

```

STACKEDBAR-CYLINDER.CS

```

public ActionResult Index()
{
    List<CylinderChartData> chartData = new List<CylinderChartData>
    {
        new CylinderChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new CylinderChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new CylinderChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
    }
}

```

```

        new CylinderChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new CylinderChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new CylinderChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new CylinderChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new CylinderChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new CylinderChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new CylinderChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class CylinderChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Series customization

The following properties can be used to customize the [Stacked Bar](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("yValue").
    DashArray("5,5").
    Opacity(0.7).
    Fill("blue").
    Border(br => br.Width(2).Color("black")).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar).
    XName("x").
    YName("yValue1").
    DashArray("5,5").
    Opacity(0.7).
    Fill("green").
    Border(br => br.Width(2).Color("black")).
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()

```


STACKEDBAR-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

100% Stacked Bar in ASP.NET MVC Charts Component

100% Stacked bar

To render a 100% stacked bar series, use series [Type](#) as [StackingBar100](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .IsIndexed(true)

```

```
)
.Title("Olympic Medal Counts - RIO").Render()
```

STACKEDBAR100.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}
```

100% Cylindrical stacked bar chart

To render a 100% cylindrical stacked bar chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
}))
```

```
.Render())
```

STACKEDBAR100-CYLINDER.CS

```
public ActionResult Index()
{
    List<CylinderChartData> chartData = new List<CylinderChartData>
    {
        new CylinderChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new CylinderChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new CylinderChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new CylinderChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new CylinderChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new CylinderChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new CylinderChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new CylinderChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new CylinderChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new CylinderChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class CylinderChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}
```

Series customization

The following properties can be used to customize the [100% Stacked Bar](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [DashArray](#) – Specifies the dashes for series.
- [ChartSeriesBorder](#) – Specifies the [Color](#) and [Width](#) of series border.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue").
    DashArray("5,5").
    Fill("blue").
    Opacity(0.7).
    Border(br => br.Width(2).Color("black")).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue1").
```

```

        DashArray("5,5").
        Fill("green").
        Opacity(0.7).
        Border(br => br.Width(2).Color("black")).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Render()

```

STACKEDBAR-100-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

Scatter in ASP.NET MVC Charts Component

Scatter

To render a scatter series, use series [Type](#) as [Scatter](#).

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
}

```

```

        Width(2).
        Add();
    })
    .Title("Olympic Medal Counts - RIO").Load("load").Render()
<script>
function load(args) {
args.chart.series[0].dataSource = [
    { 'x': 115, 'y': 57 }, { 'x': 138, 'y': 57 }, { 'x': 166, 'y': 57 }, {
    'x': 122, 'y': 57 },
    { 'x': 126, 'y': 57 }, { 'x': 130, 'y': 57 }, { 'x': 125, 'y': 57 }, {
    'x': 144, 'y': 57 },
    { 'x': 150, 'y': 57 }, { 'x': 120, 'y': 57 }, { 'x': 125, 'y': 57 }, {
    'x': 130, 'y': 57 },
    { 'x': 103, 'y': 58 }, { 'x': 116, 'y': 58 }, { 'x': 130, 'y': 58 }, {
    'x': 126, 'y': 58 },
    { 'x': 136, 'y': 58 }, { 'x': 148, 'y': 58 }, { 'x': 119, 'y': 58 }, {
    'x': 141, 'y': 58 },
    { 'x': 159, 'y': 58 }, { 'x': 120, 'y': 58 }, { 'x': 135, 'y': 58 }, {
    'x': 163, 'y': 58 },
    { 'x': 119, 'y': 59 }, { 'x': 131, 'y': 59 }, { 'x': 148, 'y': 59 }, {
    'x': 123, 'y': 59 },
    { 'x': 137, 'y': 59 }, { 'x': 149, 'y': 59 }, { 'x': 121, 'y': 59 }, {
    'x': 142, 'y': 59 },
    { 'x': 160, 'y': 59 }, { 'x': 118, 'y': 59 }, { 'x': 130, 'y': 59 }, {
    'x': 146, 'y': 59 },
    { 'x': 119, 'y': 60 }, { 'x': 133, 'y': 60 }, { 'x': 150, 'y': 60 }, {
    'x': 133, 'y': 60 },
    { 'x': 149, 'y': 60 }, { 'x': 165, 'y': 60 }, { 'x': 130, 'y': 60 }, {
    'x': 139, 'y': 60 },
    { 'x': 154, 'y': 60 }, { 'x': 118, 'y': 60 }, { 'x': 152, 'y': 60 }, {
    'x': 154, 'y': 60 },
    { 'x': 130, 'y': 61 }, { 'x': 145, 'y': 61 }, { 'x': 166, 'y': 61 }, {
    'x': 131, 'y': 61 },
    { 'x': 143, 'y': 61 }, { 'x': 162, 'y': 61 }, { 'x': 131, 'y': 61 }, {
    'x': 145, 'y': 61 },
    { 'x': 162, 'y': 61 }, { 'x': 115, 'y': 61 }, { 'x': 149, 'y': 61 }, {
    'x': 183, 'y': 61 },
    { 'x': 121, 'y': 62 }, { 'x': 139, 'y': 62 }, { 'x': 159, 'y': 62 }, {
    'x': 135, 'y': 62 },
    { 'x': 152, 'y': 62 }, { 'x': 178, 'y': 62 }, { 'x': 130, 'y': 62 }, {
    'x': 153, 'y': 62 },
    { 'x': 172, 'y': 62 }, { 'x': 114, 'y': 62 }, { 'x': 135, 'y': 62 }, {
    'x': 154, 'y': 62 },
    { 'x': 126, 'y': 63 }, { 'x': 141, 'y': 63 }, { 'x': 160, 'y': 63 }, {
    'x': 135, 'y': 63 },
    { 'x': 149, 'y': 63 }, { 'x': 180, 'y': 63 }, { 'x': 132, 'y': 63 }, {
    'x': 144, 'y': 63 },
    { 'x': 163, 'y': 63 }, { 'x': 122, 'y': 63 }, { 'x': 146, 'y': 63 }, {
    'x': 156, 'y': 63 },
    { 'x': 133, 'y': 64 }, { 'x': 150, 'y': 64 }, { 'x': 176, 'y': 64 }, {
    'x': 133, 'y': 64 },
    { 'x': 149, 'y': 64 }, { 'x': 176, 'y': 64 }, { 'x': 136, 'y': 64 }, {
    'x': 157, 'y': 64 },
    { 'x': 174, 'y': 64 }, { 'x': 131, 'y': 64 }, { 'x': 155, 'y': 64 }, {
    'x': 191, 'y': 64 },
    { 'x': 136, 'y': 65 }, { 'x': 149, 'y': 65 }, { 'x': 177, 'y': 65 }, {
    'x': 143, 'y': 65 },

```

```

    { 'x': 149, 'y': 65 }, { 'x': 184, 'y': 65 }, { 'x': 128, 'y': 65 }, {
    'x': 146, 'y': 65 },
    { 'x': 157, 'y': 65 }, { 'x': 133, 'y': 65 }, { 'x': 153, 'y': 65 }, {
    'x': 173, 'y': 65 },
    { 'x': 141, 'y': 66 }, { 'x': 156, 'y': 66 }, { 'x': 175, 'y': 66 }, {
    'x': 125, 'y': 66 },
    { 'x': 138, 'y': 66 }, { 'x': 165, 'y': 66 }, { 'x': 122, 'y': 66 }, {
    'x': 164, 'y': 66 },
    { 'x': 182, 'y': 66 }, { 'x': 137, 'y': 66 }, { 'x': 157, 'y': 66 }, {
    'x': 176, 'y': 66 },
    { 'x': 149, 'y': 67 }, { 'x': 159, 'y': 67 }, { 'x': 179, 'y': 67 }, {
    'x': 156, 'y': 67 },
    { 'x': 179, 'y': 67 }, { 'x': 186, 'y': 67 }, { 'x': 147, 'y': 67 }, {
    'x': 166, 'y': 67 },
    { 'x': 185, 'y': 67 }, { 'x': 140, 'y': 67 }, { 'x': 160, 'y': 67 }, {
    'x': 180, 'y': 67 },
    { 'x': 145, 'y': 68 }, { 'x': 155, 'y': 68 }, { 'x': 170, 'y': 68 }, {
    'x': 129, 'y': 68 },
    { 'x': 164, 'y': 68 }, { 'x': 189, 'y': 68 }, { 'x': 150, 'y': 68 }, {
    'x': 157, 'y': 68 },
    { 'x': 183, 'y': 68 }, { 'x': 144, 'y': 68 }, { 'x': 170, 'y': 68 }, {
    'x': 180, 'y': 68 }
];
}
</script>

```

SCATTER.CS

```

public IActionResult Index() {
    return View();
}

```

Series customization

The following properties can be used to customize the [Scatter](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).
- [Shape](#) – Specifies the shape of the scatter series.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter).
    XName("x").
    YName("yValue").
    Opacity(0.5).
    Fill("blue").
    Marker(mr =>
mr.Visible(false).Width(10).Height(10).Shape(Syncfusion.EJ2.Charts.ChartShape.Circle)).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter).

```

```

        XName("x").
        YName("yValue1").
        Opacity(0.5).
        Fill("red").
        Marker(mr =>
mr.Visible(false).Width(10).Height(10).Shape(Syncfusion.EJ2.Charts.ChartShape.Diamond)).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Render()

```

SCATTER-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Bubble in ASP.NET MVC Charts Component

Bubble

To render a bubble series, use series [Type](#) as [Bubble](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bubble).
    XName("x").

```

```

        YName("Y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Size("size").
        Width(2).
        Add();
    })
    .PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Title("Olympic Medal Counts - RIO").Render()

```

BUBBLE.CS

```

public ActionResult Index()
{
    List<BubbleChartData> chartData = new List<BubbleChartData>
    {
        new BubbleChartData { x= 92.2, y= 7.8, size= 1.347, text=
"China" },
        new BubbleChartData { x= 74, y= 6.5, size= 1.241, text=
"India" },
        new BubbleChartData { x= 90.4, y= 6.0, size= 0.238, text=
"Indonesia" },
        new BubbleChartData { x= 99.4, y= 2.2, size= 0.312, text=
"US" },
        new BubbleChartData { x= 88.6, y= 1.3, size= 0.197, text=
"Brazil" },
        new BubbleChartData { x= 99, y= 0.7, size= 0.0818, text=
"Germany" },
        new BubbleChartData { x= 72, y= 2.0, size= 0.0826, text=
"Egypt" },
        new BubbleChartData { x= 99.6, y= 3.4, size= 0.143, text=
"Russia" },
        new BubbleChartData { x= 99, y= 0.2, size= 0.128, text=
"Japan" },
        new BubbleChartData { x= 86.1, y= 4.0, size= 0.115, text=
"Mexico" },
        new BubbleChartData { x= 92.6, y= 6.6, size= 0.096, text=
"Philippines" },
        new BubbleChartData { x= 61.3, y= 1.45, size= 0.162, text=
"Nigeria" },
        new BubbleChartData { x= 82.2, y= 3.97, size= 0.7, text=
"Hong Kong" },
        new BubbleChartData { x= 79.2, y= 3.9, size= 0.162, text=
"Netherland" },
        new BubbleChartData { x= 72.5, y= 4.5, size= 0.7, text=
"Jordan" },
        new BubbleChartData { x= 81, y= 3.5, size= 0.21, text=
"Australia" },
        new BubbleChartData { x= 66.8, y= 3.9, size= 0.028, text=
"Mongolia" },
        new BubbleChartData { x= 78.4, y= 2.9, size= 0.231, text=
"Taiwan" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

```



```

    }
    public class BubbleChartData
    {
        public double x;
        public double y;
        public double size;
        public string text;
    }

```

Size Mapping

Size property can be used to map the size value specified in data source.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.BUbble).
    Marker(ViewBag.Marker).
    XName("x").
    YName("y").
    Size("size").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).
    Add();
}).
.PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Title("Olympic Medal Counts - RIO").Render()

```

BUBBLE-SIZE.CS

```

public ActionResult Index()
{
    List<BubbleChartData> chartData = new List<BubbleChartData>
    {
        new BubbleChartData { x= 92.2, y= 7.8, size= 1.347, text=
"China" },
        new BubbleChartData { x= 74, y= 6.5, size= 1.241, text=
"India" },
        new BubbleChartData { x= 90.4, y= 6.0, size= 0.238, text=
"Indonesia" },
        new BubbleChartData { x= 99.4, y= 2.2, size= 0.312, text=
"US" },
        new BubbleChartData { x= 88.6, y= 1.3, size= 0.197, text=
"Brazil" },
        new BubbleChartData { x= 99, y= 0.7, size= 0.0818, text=
"Germany" },
        new BubbleChartData { x= 72, y= 2.0, size= 0.0826, text=
"Egypt" },
        new BubbleChartData { x= 99.6, y= 3.4, size= 0.143, text=
"Russia" },
        new BubbleChartData { x= 99, y= 0.2, size= 0.128, text=
"Japan" },
        new BubbleChartData { x= 86.1, y= 4.0, size= 0.115, text=
"Mexico" },
    }
}

```

```

        new BubbleChartData { x= 92.6, y= 6.6, size= 0.096, text=
        "Philippines" },
        new BubbleChartData { x= 61.3, y= 1.45, size= 0.162, text=
        "Nigeria" },
        new BubbleChartData { x= 82.2, y= 3.97, size= 0.7, text=
        "Hong Kong" },
        new BubbleChartData { x= 79.2, y= 3.9, size= 0.162, text=
        "Netherland" },
        new BubbleChartData { x= 72.5, y= 4.5, size= 0.7, text=
        "Jordan" },
        new BubbleChartData { x= 81, y= 3.5, size= 0.21, text=
        "Australia" },
        new BubbleChartData { x= 66.8, y= 3.9, size= 0.028, text=
        "Mongolia" },
        new BubbleChartData { x= 78.4, y= 2.9, size= 0.231, text=
        "Taiwan" }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class BubbleChartData
{
    public double x;
    public double y;
    public double size;
    public string text;
}

```

Series customization

The following properties can be used to customize the [Bubble](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Bubble).
    XName("x").
    YName("y").
    Opacity(0.7).
    Fill("blue").
    Size("size").
    DataSource(ViewBag.dataSource).
    Add();
})
.Render()

```

BUBBLE-SERIES.CS

```

public ActionResult Index()
{
    List<BubbleChartData> chartData = new List<BubbleChartData>

```

```

        {
            new BubbleChartData { x= 92.2, y= 7.8, size= 1.347, text=
"China" },
            new BubbleChartData { x= 74, y= 6.5, size= 1.241, text=
"India" },
            new BubbleChartData { x= 90.4, y= 6.0, size= 0.238, text=
"Indonesia" },
            new BubbleChartData { x= 99.4, y= 2.2, size= 0.312, text=
"US" },
            new BubbleChartData { x= 88.6, y= 1.3, size= 0.197, text=
"Brazil" },
            new BubbleChartData { x= 99, y= 0.7, size= 0.0818, text=
"Germany" },
            new BubbleChartData { x= 72, y= 2.0, size= 0.0826, text=
"Egypt" },
            new BubbleChartData { x= 99.6, y= 3.4, size= 0.143, text=
"Russia" },
            new BubbleChartData { x= 99, y= 0.2, size= 0.128, text=
"Japan" },
            new BubbleChartData { x= 86.1, y= 4.0, size= 0.115, text=
"Mexico" },
            new BubbleChartData { x= 92.6, y= 6.6, size= 0.096, text=
"Philippines" },
            new BubbleChartData { x= 61.3, y= 1.45, size= 0.162, text=
"Nigeria" },
            new BubbleChartData { x= 82.2, y= 3.97, size= 0.7, text=
"Hong Kong" },
            new BubbleChartData { x= 79.2, y= 3.9, size= 0.162, text=
"Netherland" },
            new BubbleChartData { x= 72.5, y= 4.5, size= 0.7, text=
"Jordan" },
            new BubbleChartData { x= 81, y= 3.5, size= 0.21, text=
"Australia" },
            new BubbleChartData { x= 66.8, y= 3.9, size= 0.028, text=
"Mongolia" },
            new BubbleChartData { x= 78.4, y= 2.9, size= 0.231, text=
"Taiwan" }
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class BubbleChartData
    {
        public double x;
        public double y;
        public double size;
        public string text;
    }

```

See Also

- [Data Label](#)
- [Tooltip](#)

Polar in ASP.NET MVC Charts Component

Polar

To render a polar series, use series [Type](#) as [Polar](#).

Draw Types

Polar [drawType](#) property is used to change the series plotting type to line, column, area, range column, spline, scatter, stacking area and stacking column. The default value of [drawType](#) is [Line](#).

Line

To render a line draw type, use series [DrawType](#) as [Line](#). [IsClosed](#) property specifies whether to join start and end point of a line series used in polar chart to form a closed path. Default value of [isClosed](#) is true.

CSHTML

```
@ (Html.EJS () .Chart ("container")
    .Series (sr =>
    {
        sr.Type (Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
            .DrawType (Syncfusion.EJ2.Charts.ChartDrawType.Line)
            .Name ("Warmest")
            .XName ("x")
            .YName ("y")
            .Width (2)
            .Marker (ViewBag.marker1)
            .DataSource (ViewBag.dataSource) .Add ();
    })
    .PrimaryXAxis (xaxis =>
        xaxis.ValueType (Syncfusion.EJ2.Charts.ValueType.Category)
            .LabelPlacement (Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
            .Interval (1)
            .Coefficient (100)
            .Title ("Months")
    )
    .PrimaryYAxis (yaxis =>
        yaxis.Title ("Temperature (Celsius)") .LabelFormat ("{value}°C")
            .Interval (10)
            .Minimum (-25)
            .Maximum (25)
    )
    .EdgeLabelPlacement (Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title ("Alaska Weather Statistics - 2016")
    .Tooltip (tp => tp.Enable (true)) .Load ("load") .Render ()
)
```

POLAR-LINE.CS

```
public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData { x=2005, y = 28 },
        new PolarData { x=2006, y = 25 },
        new PolarData { x=2007, y = 26 },
        new PolarData { x=2008, y = 27 },
        new PolarData { x=2009, y = 32 },
    }
}
```

```

        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
}

```

Spline

To render a spline line draw type, use series [DrawType](#) as [Spline](#).

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
            .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Spline)
            .Name("Warmest")
            .XName("x")
            .YName("y")
            .Width(2)
            .Marker(ViewBag.marker1)
            .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
            .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
            .Interval(1)
            .Coefficient(100)
            .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
            .Interval(10)
            .Minimum(-25)
            .Maximum(25)
    )
    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
)

```

POLAR-SPLINE.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2005, y = 28 },
        new PolarData{ x=2006, y = 25 },
    }
}

```

```

        new PolarData{ x=2007, y = 26 },
        new PolarData{ x=2008, y = 27 },
        new PolarData{ x=2009, y = 32 },
        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PolarData
{
    public double x;
    public double y;
}

```

Area

To render a area line draw type, use series [DrawType](#) as [Area](#).

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Area)
        .Name("Warmest")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
        .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
        .Interval(10)
        .Minimum(-25)
        .Maximum(25)
    )
    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
)

```

POLAR-AREA.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>

```

```

        {
            new PolarData{ x=2005, y = 28 },
            new PolarData{ x=2006, y = 25 },
            new PolarData{ x=2007, y = 26 },
            new PolarData{ x=2008, y = 27 },
            new PolarData{ x=2009, y = 32 },
            new PolarData{ x=2010, y = 35 },
            new PolarData{ x=2011, y = 30 }
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class PolarData
    {
        public double x;
        public double y;
    }

```

Stacked Area

To render a stacked area draw type, use series [DrawType](#) as [StackingArea](#).

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.StackingArea)
        .Name("2013")
        .XName("x")
        .YName("y")
        .Width(2)
        .DataSource(ViewBag.dataSource).Add();
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.StackingArea)
        .Name("2014")
        .XName("x")
        .YName("y1")
        .Width(2)
        .DataSource(ViewBag.dataSource).Add();
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.StackingArea)
        .Name("2015")
        .XName("x")
        .YName("y2")
        .Width(2)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}M")
    )

```

```

        .Interval(5000)
        .Minimum(20000)
        .Maximum(0)
    )
    .Title("GDP, Current Prices (in Billions)")
    .Tooltip(tp => tp.Enable(true))
    .Load("load").Render()
)

```

POLAR-STACKEDAREA.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PolarData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Column

To render a column draw type, use series [DrawType](#) as [Column](#).

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Column)
        .Name("Warmest")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
)

```



```

        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
        .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
        .Interval(10)
        .Minimum(-25)
        .Maximum(25)
    )
    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
    )

```

POLAR-COLUMN.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x="2005", y= 28 },
        new PolarData{ x="2006", y= 25 },
        new PolarData{ x="2007", y= 26 },
        new PolarData{ x="2008", y= 27 },
        new PolarData{ x="2009", y= 32 },
        new PolarData{ x="2010", y= 35 },
        new PolarData{ x="2011", y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PolarData
{
    public string x;
    public double y;
}

```

Stacked Column

To render a stacked column draw type, use series [DrawType](#) as [StackingColumn](#).

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.StackingColumn)
        .Name("2013")
        .XName("x")
        .YName("y")
        .Width(2)
        .DataSource(ViewBag.dataSource).Add();
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
    }
)

```

```

        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.StackingColumn)
        .Name("2014")
        .XName("x")
        .YName("y1")
        .Width(2)
        .DataSource(ViewBag.dataSource).Add();
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
    .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.StackingColumn)
    .Name("2015")
    .XName("x")
    .YName("y2")
    .Width(2)
    .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}M")
        .Interval(5000)
        .Minimum(20000)
        .Maximum(0)
    )
    .Title("GDP, Current Prices (in Billions)")
    .Tooltip(tp => tp.Enable(true))
    .Load("load").Render()
)

```

POLAR-STACKEDCOLUMN.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2000, y= 0.61, y1= 0.03, y2= 0.48},
        new PolarData{ x=2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new PolarData{ x=2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new PolarData{ x=2003, y= 1, y1= 0.09, y2= 0.61 },
        new PolarData{ x=2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new PolarData{ x=2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new PolarData{ x=2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new PolarData{ x=2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new PolarData{ x=2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new PolarData{ x=2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PolarData
{
    public double x;
    public double y;
    public double y1;

```

```

        public double y2;
    }

```

Range Column

To render a range column draw type, use series [DrawType](#) as [RangeColumn](#).

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.RangeColumn)
        .Name("Germany")
        .XName("x")
        .High("high")
        .Low("low")
        .Width(2)
        .Border(ViewBag.border)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
        .StartAngle(90)
        .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}°C")
        .Interval(5)
        .Minimum(0)
        .Maximum(20)
    )
    .Title("Maximum and Minimum Temperature")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").Render()
)

```

POLAR-RANGECOLUMN.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x="Jan", low= 0.7, high= 6.1 },
        new PolarData{ x="Feb", low=1.3, high=6.3 },
        new PolarData{ x="Mar", low= 1.9, high= 8.5 },
        new PolarData{ x="Apr", low= 3.1, high= 10.8 },
        new PolarData{ x="May", low= 5.7, high= 14.40 },
        new PolarData { x= "Jun", low= 8.4, high= 16.90 },
        new PolarData { x= "Jul", low= 10.6,high= 19.20 },
        new PolarData { x= "Aug", low= 10.5,high= 18.9 },
        new PolarData { x= "Sep", low= 8.5, high= 16.1 },
    }
}

```

```

        new PolarData { x= "Oct", low= 6.0, high= 12.5 },
        new PolarData { x= "Nov", low= 1.5, high= 6.9 },
        new PolarData { x= "Dec", low= 5.1, high= 12.1 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PolarData
{
    public string x;
    public double low;
    public double high;
}

```

Scatter

To render a scatter draw type, use series [DrawType](#) as [Scatter](#).

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Scatter)
        .Name("2015")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Interval(2)
        .Minimum(0)
        .Maximum(8)
    )
    .Title("Real GDP Growth")
    .Tooltip(tp => tp.Enable(true))
    .Load("load").Render()
)

```

POLAR-SCATTER.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x="2005", y= 28 },
        new PolarData{ x="2006", y= 25 },
        new PolarData{ x="2007", y= 26 },
    }
}

```

```

        new PolarData{ x="2008", y= 27 },
        new PolarData{ x="2009", y= 32 },
        new PolarData{ x="2010", y= 35 },
        new PolarData{ x="2011", y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class PolarData
{
    public string x;
    public double y;
}

```

Series customization

Start Angle

You can customize the start angle of the polar series using [StartAngle](#) property. By default, **StartAngle** is 0 degree.

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Column)
        .Name("Warmest")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .StartAngle(270)
        .Coefficient(100)
        .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
        .Interval(10)
        .Minimum(-25)
        .Maximum(25)
    )
    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
)

```

START-ANGLE.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2005, y = 28 },
        new PolarData{ x=2006, y = 25 },
        new PolarData{ x=2007, y = 26 },
        new PolarData{ x=2008, y = 27 },
        new PolarData{ x=2009, y = 32 },
        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
}

```

Coefficient in axis

You can customize the radius of the polar series using [Coefficient](#) property. By default, **Coefficient** is 100.

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Area)
        .Name("Warmest")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(40)
        .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
        .Interval(10)
        .Minimum(-25)
        .Maximum(25)
    )
    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()

```

```
)
```

CO-EFFICIENT.CS

```
public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2005, y = 28 },
        new PolarData{ x=2006, y = 25 },
        new PolarData{ x=2007, y = 26 },
        new PolarData{ x=2008, y = 27 },
        new PolarData{ x=2009, y = 32 },
        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
}
```

See Also

- [Data Label](#)
- [Tooltip](#)

Radar in ASP.NET MVC Charts Component

*Radar*To render a radar series, use series [Type](#) as [Radar](#).**CSHTML**

```
@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Radar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Line)
        .Name("Warmest")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(100)
        .Title("Months")
    )
)
```

```

    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
            .Interval(10)
            .Minimum(-25)
            .Maximum(25)

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
    )

```

RADAR-LINE.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2005, y = 28 },
        new PolarData{ x=2006, y = 25 },
        new PolarData{ x=2007, y = 26 },
        new PolarData{ x=2008, y = 27 },
        new PolarData{ x=2009, y = 32 },
        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
}

```

Series customization

Start Angle

You can customize the start angle of the radar series using [StartAngle](#) property. By default, **StartAngle** is 0 degree.

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
        .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Column)
        .Name("Warmest")
        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })

```



```

        .PrimaryXAxis(xaxis =>
            xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
            .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
            .Interval(1)
            .StartAngle(270)
            .Coefficient(100)
            .Title("Months")
        )
        .PrimaryYAxis(yaxis =>
            yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
            .Interval(10)
            .Minimum(-25)
            .Maximum(25)

        .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
        )
        .Title("Alaska Weather Statistics - 2016")
        .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
    )

```

START-ANGLE.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2005, y = 28 },
        new PolarData{ x=2006, y = 25 },
        new PolarData{ x=2007, y = 26 },
        new PolarData{ x=2008, y = 27 },
        new PolarData{ x=2009, y = 32 },
        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
}

```

Coefficient in axis

You can customize the radius of the radar series using [Coefficient](#) property. By default, **Coefficient** is 100.

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
        {
            sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Polar)
            .DrawType(Syncfusion.EJ2.Charts.ChartDrawType.Area)
            .Name("Warmest")

```

```

        .XName("x")
        .YName("y")
        .Width(2)
        .Marker(ViewBag.marker1)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
        .Interval(1)
        .Coefficient(40)
        .Title("Months")
    )
    .PrimaryYAxis(yaxis =>
        yaxis.Title("Temperature (Celsius)").LabelFormat("{value}°C")
        .Interval(10)
        .Minimum(-25)
        .Maximum(25)

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    )
    .Title("Alaska Weather Statistics - 2016")
    .Tooltip(tp=>tp.Enable(true)).Load("load").Render()
    )

```

CO-EFFICIENT.CS

```

public IActionResult Index()
{
    List<PolarData> chartData = new List<PolarData>
    {
        new PolarData{ x=2005, y = 28 },
        new PolarData{ x=2006, y = 25 },
        new PolarData{ x=2007, y = 26 },
        new PolarData{ x=2008, y = 27 },
        new PolarData{ x=2009, y = 32 },
        new PolarData{ x=2010, y = 35 },
        new PolarData{ x=2011, y = 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class PolarData
{
    public double x;
    public double y;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Hilo in ASP.NET MVC Charts Component

Hilo

Hilo Series illustrates the price movements in stock using the high and low values. To render a Hilo series, use series [Type](#) as [Hilo](#).

Hilo series requires 3 fields (x, high and low) to show the high and low price in the stock.

CSHTML

```
<script src="chart/series/hilo/financial-data.js"></script>
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Hilo).
    XName("x").
    High("high").
    Low("low").
    DataSource("returnValue").
    Name("Apple Inc").
    Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .CrosshairTooltip(ViewBag.crosshairTooltip)
)
.ChartArea(area => area.Border(ViewBag.ChartBorder))
.Crosshair(cr =>
    cr.Enable(true).LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Line(ViewBag.line)
)
.LegendSettings(lg => lg.Visible(false))
.Tooltip(tt => tt.Enable(true).Shared(true))
.Title("AAPL Historical").Load("load").Render()
<script>
var dataSource = window.chartData;
var date1 = new Date('2017-01-01');
var returnValue = dataSource.filter(filterValue);
function filterValue(value) {
    return value.x >= date1;
}
</script>
```

HILO.CS

```
public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", low= 87, high= 200 },
        new Data{ x= "Feb", low= 45, high= 135 },
        new Data{ x= "Mar", low= 19, high= 85 },
        new Data{ x= "Apr", low= 31, high= 108 },
        new Data{ x= "May", low= 27, high= 80 },
        new Data{ x= "June", low= 84, high= 130 },
        new Data{ x= "Jul", low= 77, high= 150 },
        new Data{ x= "Aug", low= 54, high= 125 },
        new Data{ x= "Sep", low= 60, high= 155 },
    }
```

```

        new Data{ x= "Oct", low= 60, high= 180 },
        new Data{ x= "Nov", low= 88, high= 180 },
        new Data{ x= "Dec", low= 84, high= 230 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double y;
    public double high;
    public double low;
}

```

Series customization

The following properties can be used to customize the [Hilo](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of [Fill](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Hilo).
    XName("x").
    High("high").
    Low("low").
    DataSource(ViewBag.dataSource).
    Opacity(0.5).
    Fill("blue").
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()

```

HILO-SERIES.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", low= 87, high= 200 },
        new Data{ x= "Feb", low= 45, high= 135 },
        new Data{ x= "Mar", low= 19, high= 85 },
        new Data{ x= "Apr", low= 31, high= 108 },
        new Data{ x= "May", low= 27, high= 80 },
        new Data{ x= "June", low= 84, high= 130 },
        new Data{ x= "Jul", low= 77, high= 150 },
        new Data{ x= "Aug", low= 54, high= 125 },
        new Data{ x= "Sep", low= 60, high= 155 },
        new Data{ x= "Oct", low= 60, high= 180 },
    }
}

```

```

        new Data{ x= "Nov", low= 88, high= 180 },
        new Data{ x= "Dec", low= 84, high= 230 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double high;
    public double low;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

High Low Open Close in ASP.NET MVC Charts Component

High Low Open Close

HiloOpenClose series is used to represent the low, high, open and closing values over time. To render a HiloOpenClose series, use series [Type](#) as [HiloOpenClose](#).

HiloOpenClose series requires 5 fields (x, high, low, open and close) to show the high, low, open and close price values in the stock.

CSHTML

```

<script src="chart/series/hilo/financial-data.js"></script>
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.HiloOpenClose).
    XName("x").
    High("high").
    Low("low").
    Close("close").
    Open("open").
    DataSource("returnValue").
    Name("Warmest").
    Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .CrosshairTooltip(ViewBag.crosshairTooltip)
)
.PrimaryYAxis(py =>
    py.Minimum(100)
    .Maximum(180)
    .Interval(20)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Title("Price")
    .LabelFormat("${value}")
)
)

```

```

        .ChartArea(area => area.Border(ViewBag.ChartBorder))
        .Crosshair(cr =>
cr.Enable(true).Line(ViewBag.line).LineType(Syncfusion.EJ2.Charts.LineType.V
ertical))
        .LegendSettings(lg => lg.Visible(false))
        .Tooltip(tt => tt.Enable(true).Shared(true))
        .Title("AAPL Historical").Load("load").Render()
<script>
    var dataSource = window.chartData;
    var date1 = new Date('2017-01-01');
    var returnValue = dataSource.filter(filterValue);
    function filterValue(value) {
        return value.x >= date1;
    }
</script>

```

HILO-OPENCLOSE.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", open= 120, high= 160, low= 100, close=
140 },
        new Data{ x= "Feb", open= 150, high= 190, low= 130, close=
170 },
        new Data{ x= "Mar", open= 130, high= 170, low= 110, close=
150 },
        new Data{ x= "Apr", open= 160, high= 180, low= 120, close=
140 },
        new Data{ x= "May", open= 150, high= 170, low= 110, close=
130 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double y;
    public double high;
    public double low;
    public double open;
    public double close;
}

```

Series customization

In HiloOpenClose series, [BullFillColor](#) is used to fill the segment when the open value is greater than the close value and [BearFillColor](#) is used to fill the segment when the open value is less than the close value.

By default, bullFillColor is set as red and bearFillColor is set as green.

CSHTML

```

<script src="chart/series/hilo/financial-data.js"></script>
@Html.EJS().Chart("container").Series(series =>

```

```

{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.HiloOpenClose)
    .XName("x")
    .High("high")
    .Low("low")
    .Close("close")
    .Open("open")
    .DataSource("returnValue")
    .BearFillColor("#2ecd71")
    .BullFillColor("#e74c3d")
    .Name("Warmest")
    .Add();
}) .PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .CrosshairTooltip(ViewBag.crosshairTooltip)
) .PrimaryYAxis(py =>
    py.Minimum(100)
    .Maximum(180)
    .Interval(20)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle)
    .Title("Price")
    .LabelFormat("${value}")
) .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Crosshair(cr =>
cr.Enable(true).Line(ViewBag.line).LineType(Syncfusion.EJ2.Charts.LineType.V
ertical))
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true).Shared(true))
    .Title("AAPL Historical").Load("load").Render()
}
<script>
    var dataSource = window.chartData;
    var date1 = new Date('2017-01-01');
    var returnValue = dataSource.filter(filterValue);
    function filterValue(value) {
        return value.x >= date1;
    }
</script>

```

CUSTOM-OPENCLOSE.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", open= 120, high= 160, low= 100, close=
140 },
        new Data{ x= "Feb", open= 150, high= 190, low= 130, close=
170 },
        new Data{ x= "Mar", open= 130, high= 170, low= 110, close=
150 },
        new Data{ x= "Apr", open= 160, high= 180, low= 120, close=
140 },
    }
}

```

```

130         new Data{ x= "May", open= 150, high= 170, low= 110, close=
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double y;
    public double high;
    public double low;
    public double open;
    public double close;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Candle in ASP.NET MVC Charts Component

Candle

Candle series are similar to Hilo Open Close series, are used to represent the low, high, open and closing price over time. To render a candle series, use series [Type](#) as [Candle](#).

CSHTML

```

<script src="chart/series/hilo/financial-data.js"></script>
@Html.EJS().Chart("container").Rows(rows =>
{
    rows.Height("30%").Add();
    rows.Height("70%").Add();
}).Axes(ax =>
{
    ax.Name("secondary").
    OpposedPosition(true).
    Minimum(100).
    Maximum(180).
    RowIndex(1).
    PlotOffset(30).
    Title("Price").
    LabelFormat("${value}").
    Interval(20).
    MajorTickLines(ViewBag.majorTickLines).
    LineStyle(ViewBag.lineStyle).Add();
}
    ).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("volume").Name("Volume").DataSource("returnValue").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").High("h

```



```

igh").Low("low").Close("close").Open("open").DataSource("returnValue").YAxis
Name("secondary").Name("Apple Inc").Add();
    }).PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .MajorGridLines(ViewBag.majorGridLines)
        .CrosshairTooltip(ViewBag.crosshairTooltip)
    ).PrimaryYAxis(py =>
        py.Minimum(500000000)
        .Maximum(1300000000)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic)
        .OpposedPosition(true)
        .Title("Volume")
        .LineStyle(ViewBag.lineStyle)
    ).Crosshair(cr =>
cr.Enable(true).LineType(Syncfusion.EJ2.Charts.LineType.Vertical))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .LegendSettings(lg => lg.Visible(true))
    .Tooltip(tt => tt.Enable(true).Shared(true))
    .Title("AAPL Historical").Width("60%").Load("load").Render()
}
<script>
    var dataSource = window.chartData;
    var date1 = new Date('2017-01-01');
    var returnValue = dataSource.filter(filterValue);
    function filterValue(value) {
        return value.x >= date1;
    }
</script>

```

CANDLE.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", open= 120, high= 160, low= 100, close=
140 },
        new Data{ x= "Feb", open= 150, high= 190, low= 130, close=
170 },
        new Data{ x="Mar", open= 130, high= 170, low= 110, close=
150 },
        new Data{ x= "Apr", open= 160, high= 180, low= 120, close=
140 },
        new Data{ x= "May", open= 150, high= 170, low= 110, close=
130 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double y;
    public double high;
    public double low;
}

```

```

        public double open;
        public double close;
    }

```

Hollow Candles

Candle charts allow to visually compare the current price with previous price by customizing the appearance.

Candles are filled/left as hollow based on the following criteria.

<!-- markdownlint-disable MD033 -->

States	Description
Filled	candle sticks are filled when the close value is lesser than the open value
Unfilled	candle sticks are unfilled when the close value is greater than the open value

The color of the candle will be defined by comparing with previous values.

- Bear color will be applied when the current closing value is greater than the previous closing value.
- Bull color will be applied when the current closing value is less than the previous closing value.

By default, bullFillColor is set as red and bearFillColor is set as green.

Solid Candles

[EnableSolidCandles](#) is used to enable/disable the solid candles. By default is set to be false. The fill color of the candle will be defined by its opening and closing values.

[BearFillColor](#) will be applied when the opening value is less than the closing value. [BullFillColor](#) will be applied when the opening value is greater than closing value.

CSHTML

```

<script src="chart/series/hilo/financial-data.js"></script>
@Html.EJS().Chart("container").Rows(rows =>
{
    rows.Height("30%").Add();
    rows.Height("70%").Add();
}) .Axes(ax =>
{
    ax.Name("secondary") .
        OpposedPosition(true) .
        Minimum(100) .
        Maximum(180) .
        RowIndex(1) .
        PlotOffset(30) .
        Title("Price") .
        LabelFormat("${value}") .
        Interval(20) .
        MajorTickLines(ViewBag.majorTickLines) .
        LineStyle(ViewBag.lineStyle) .
        Add();
}

```

```

    }

    ).Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
        .XName("x")
        .YName("volume")
        .Name("Volume")
        .DataSource("returnValue").
        Add();
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle)
        .XName("x")
        .High("high")
        .Low("low")
        .Close("close")
        .Open("open")
        .BearFillColor("#2ecd71")
        .BullFillColor("#e74c3d")
        .DataSource("returnValue")
        .YAxisName("secondary")
        .Name("Apple Inc")
        .Add();
    }).PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .MajorGridLines(ViewBag.majorGridLines)
        .CrosshairTooltip(ViewBag.crosshairTooltip)
    )
    .PrimaryYAxis(py =>
        py.Minimum(500000000)
        .Maximum(1300000000)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic)
        .OpposedPosition(true)
        .Title("Volume")
        .LineStyle(ViewBag.lineStyle)
    )
    .Crosshair(cr =>
cr.Enable(true).LineType(Syncfusion.EJ2.Charts.LineType.Vertical))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .LegendSettings(lg => lg.Visible(true))
    .Tooltip(tt => tt.Enable(true).Shared(true))
    .Title("AAPL Historical").Width("60%").Load("load").Render()

<script>
    var dataSource = window.chartData;
    var date1 = new Date('2017-01-01');
    var returnValue = dataSource.filter(filterValue);
    function filterValue(value) {
        return value.x >= date1;
    }
</script>

```

SOLID-CANDLES.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {

```

```

140     new Data{ x= "Jan", open= 120, high= 160, low= 100, close=
170     },
    new Data{ x= "Feb", open= 150, high= 190, low= 130, close=
150     },
    new Data{ x= "Mar", open= 130, high= 170, low= 110, close=
140     },
    new Data{ x= "Apr", open= 160, high= 180, low= 120, close=
130     },
    new Data{ x= "May", open= 150, high= 170, low= 110, close=
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double y;
    public double high;
    public double low;
    public double open;
    public double close;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Box and Whisker in ASP.NET MVC Charts Component

Box and whisker

To render a box and whisker chart, use series [Type](#) as [BoxAndWhisker](#). The field y requires n number of data or it should contains minimum of five values to plot a segment.

CSHTML

```

@(Html.EJS().Chart("container").Series(
    series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.BoxAndWhisker).
    Name("Department").
    DataSource(ViewBag.dataSource).
    XName("xValue").
    YName("yValue").
    Marker(ViewBag.marker).Add();
}).PrimaryXAxis(
    xAxis =>
        xAxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Trim)

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
        .MajorGridLines(ViewBag.gridlines)
    ).PrimaryYAxis(

```

```

        yAxis =>
        {
            yAxis.Minimum(10).
            Maximum(60).
            Interval(10).
            Title("Age").
            MajorGridLines(ViewBag.gridlines).
            MajorTickLines(ViewBag.ticklines)
        }.Title("Employee Age Group in Various Department")
        .LegendSettings(
            legend => {
                legend.Visible(false);
            }).Tooltip(
            tooltip =>
            {
                tooltip.Enable(true);
            }).ChartArea(area => area.Border(ViewBag.ChartBorder)).Render()
    )

```

BOX.CS

```

public ActionResult Index()
{
    Double[] y1 = { 22, 22, 23, 25, 25, 25, 26, 27, 27, 28, 28, 29,
30, 32, 34, 32, 34, 36, 35, 38 };
    Double[] y2 = { 22, 33, 23, 25, 26, 28, 29, 30, 34, 33, 32, 31,
50 };
    Double[] y3 = { 22, 24, 25, 30, 32, 34, 36, 38, 39, 41, 35, 36,
40, 56 };
    Double[] y4 = { 26, 27, 28, 30, 32, 34, 35, 37, 35, 37, 45 };
    Double[] y5 = { 26, 27, 29, 32, 34, 35, 36, 37, 38, 39, 41, 43,
58 };
    Double[] y6 = { 27, 26, 28, 29, 29, 29, 32, 35, 32, 38, 53 };
    Double[] y7 = { 21, 23, 24, 25, 26, 27, 28, 30, 34, 36, 38 };
    Double[] y8 = { 26, 28, 29, 30, 32, 33, 35, 36, 52 };
    Double[] y9 = { 28, 29, 30, 31, 32, 34, 35, 36 };
    List<BoxAndWhiskerChartData> chartData = new
List<BoxAndWhiskerChartData>
    {
        new BoxAndWhiskerChartData { xValue = "Development", yValue
= y1 },
        new BoxAndWhiskerChartData { xValue = "Testing", yValue =
y2},
        new BoxAndWhiskerChartData { xValue = "HR", yValue = y3 },
        new BoxAndWhiskerChartData { xValue = "Finance", yValue = y4
},
        new BoxAndWhiskerChartData { xValue = "R&D", yValue = y5 },
        new BoxAndWhiskerChartData { xValue = "Sales", yValue = y6
},
        new BoxAndWhiskerChartData { xValue = "Inventory", yValue =
y7 },
        new BoxAndWhiskerChartData { xValue = "Graphics", yValue =
y8 },
        new BoxAndWhiskerChartData { xValue = "Training", yValue =
y9 },
    };
    ViewBag.dataSource = chartData;
}

```

```

        return View();
    }
    public class BoxAndWhiskerChartData
    {
        public string xValue;
        public double[] yValue;
    }

```

Box Plot

You can change the rendering mode of the Box and Whisker series using the `BoxPlotMode` property. The default `boxPlotMode` is `Exclusive`. The other `boxPlotMode` available are `Inclusive` and `Normal`.

To render a box and whisker chart, use series `Type` as `BoxAndWhisker`. The field `y` requires a number of data or it should contain a minimum of five values to plot a segment.

CSHTML

```

@(Html.EJS().Chart("container").Series(
    series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.BoxAndWhisker).
            Name("Department").
            BoxPlotMode("Normal").
            DataSource(ViewBag.dataSource).
            XName("xValue").
            YName("yValue").
            Marker(ViewBag.marker).Add();
    }).PrimaryXAxis(
    xAxis =>
        xAxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Trim)

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
        .MajorGridLines(ViewBag.gridlines)
    ).PrimaryYAxis(
    yAxis =>
        yAxis.Minimum(10).
        Maximum(60).
        Interval(10).
        Title("Age").
        MajorGridLines(ViewBag.gridlines).
        MajorTickLines(ViewBag.ticklines)
    ).Title("Employee Age Group in Various Department")
    .LegendSettings(
        legend => {
            legend.Visible(false);
        }).Tooltip(
    tooltip =>
    {
        tooltip.Enable(true);
    }).ChartArea(area => area.Border(ViewBag.ChartBorder)).Render()
)

```

BOX-PLOT.CS

```

public ActionResult Index()
{
    Double[] y1 = { 22, 22, 23, 25, 25, 25, 26, 27, 27, 28, 28, 29,
30, 32, 34, 32, 34, 36, 35, 38 };
    Double[] y2 = { 22, 33, 23, 25, 26, 28, 29, 30, 34, 33, 32, 31,
50 };
    Double[] y3 = { 22, 24, 25, 30, 32, 34, 36, 38, 39, 41, 35, 36,
40, 56 };
    Double[] y4 = { 26, 27, 28, 30, 32, 34, 35, 37, 35, 37, 45 };
    Double[] y5 = { 26, 27, 29, 32, 34, 35, 36, 37, 38, 39, 41, 43,
58 };
    Double[] y6 = { 27, 26, 28, 29, 29, 29, 32, 35, 32, 38, 53 };
    Double[] y7 = { 21, 23, 24, 25, 26, 27, 28, 30, 34, 36, 38 };
    Double[] y8 = { 26, 28, 29, 30, 32, 33, 35, 36, 52 };
    Double[] y9 = { 28, 29, 30, 31, 32, 34, 35, 36 };
    List<BoxAndWhiskerChartData> chartData = new
List<BoxAndWhiskerChartData>
    {
        new BoxAndWhiskerChartData { xValue = "Development", yValue
= y1 },
        new BoxAndWhiskerChartData { xValue = "Testing", yValue =
y2},
        new BoxAndWhiskerChartData { xValue = "HR", yValue = y3 },
        new BoxAndWhiskerChartData { xValue = "Finance", yValue = y4
},
        new BoxAndWhiskerChartData { xValue = "R&D", yValue = y5 },
        new BoxAndWhiskerChartData { xValue = "Sales", yValue = y6
},
        new BoxAndWhiskerChartData { xValue = "Inventory", yValue =
y7 },
        new BoxAndWhiskerChartData { xValue = "Graphics", yValue =
y8 },
        new BoxAndWhiskerChartData { xValue = "Training", yValue =
y9 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class BoxAndWhiskerChartData
{
    public string xValue;
    public double[] yValue;
}

```

Show mean

In Box and Whisker series **ShowMean** property is used to show the box and whisker average value. The default value of **ShowMean** is false.

CSHTML

```

@(Html.EJS().Chart("container").Series(
    series =>
    {

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.BoxAndWhisker).
    Name("Department").
    BoxPlotMode("Normal").
    ShowMean("false").
    DataSource(ViewBag.dataSource).
    XName("xValue").
    YName("yValue").
    Marker(ViewBag.marker).Add();
}).PrimaryXAxis(
    xAxis =>
        xAxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Trim)

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.gridlines)
).PrimaryYAxis(
    yAxis =>
        yAxis.Minimum(10).
        Maximum(60).
        Interval(10).
        Title("Age").
        MajorGridLines(ViewBag.gridlines).
        MajorTickLines(ViewBag.ticklines)
).Title("Employee Age Group in Various Department")
.LegendSettings(
    legend => {
        legend.Visible(false);
    }).Tooltip(
    tooltip =>
    {
        tooltip.Enable(true);
    }).ChartArea(area => area.Border(ViewBag.ChartBorder)).Render()
)

```

BOX-MEAN.CS

```

public ActionResult Index()
{
    Double[] y1 = { 22, 22, 23, 25, 25, 25, 26, 27, 27, 28, 28, 29,
30, 32, 34, 32, 34, 36, 35, 38 };
    Double[] y2 = { 22, 33, 23, 25, 26, 28, 29, 30, 34, 33, 32, 31,
50 };
    Double[] y3 = { 22, 24, 25, 30, 32, 34, 36, 38, 39, 41, 35, 36,
40, 56 };
    Double[] y4 = { 26, 27, 28, 30, 32, 34, 35, 37, 35, 37, 45 };
    Double[] y5 = { 26, 27, 29, 32, 34, 35, 36, 37, 38, 39, 41, 43,
58 };
    Double[] y6 = { 27, 26, 28, 29, 29, 29, 32, 35, 32, 38, 53 };
    Double[] y7 = { 21, 23, 24, 25, 26, 27, 28, 30, 34, 36, 38 };
    Double[] y8 = { 26, 28, 29, 30, 32, 33, 35, 36, 52 };
    Double[] y9 = { 28, 29, 30, 31, 32, 34, 35, 36 };
    List<BoxAndWhiskerChartData> chartData = new
List<BoxAndWhiskerChartData>
{

```



```

= y1 },
    new BoxAndWhiskerChartData { xValue = "Development", yValue =
y2},
    new BoxAndWhiskerChartData { xValue = "Testing", yValue =
    new BoxAndWhiskerChartData { xValue = "HR", yValue = y3 },
    new BoxAndWhiskerChartData { xValue = "Finance", yValue = y4
    },
    new BoxAndWhiskerChartData { xValue = "R&D", yValue = y5 },
    new BoxAndWhiskerChartData { xValue = "Sales", yValue = y6
    },
    new BoxAndWhiskerChartData { xValue = "Inventory", yValue =
y7 },
    new BoxAndWhiskerChartData { xValue = "Graphics", yValue =
y8 },
    new BoxAndWhiskerChartData { xValue = "Training", yValue =
y9 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class BoxAndWhiskerChartData
{
    public string xValue;
    public double[] yValue;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Waterfall in ASP.NET MVC Charts Component

Waterfall

Waterfall chart helps to understand the cumulative effect of the sequentially introduced positive and negative values. To render a Waterfall series, use series [Type](#) as [Waterfall](#). [IntermediateSumIndexes](#) property of waterfall is used to represent the in between the sum values and [SumIndexes](#) is used to represent the cumulative sum values.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Width(2).
    XName("xValue").
    YName("yValue").
    IntermediateSumIndexes(ViewBag.intermediateSumIndexes).
    SumIndexes(ViewBag.sumIndexes).
    Marker(ViewBag.marker).
    Name("USA").
    ColumnWidth(0.9).
    Type(Syncfusion.EJ2.Charts.ChartSeriesType.Waterfall).
    DataSource(ViewBag.dataSource).
    Add();
})

```

```

.ChartArea(area => area.Border(ViewBag.ChartBorder))
.PrimaryXAxis(py =>
    py.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .MajorGridLines(ViewBag.majorGridLines)
    .PlotOffset(20)
)
.PrimaryYAxis(py =>
    py.Minimum(0)
    .Maximum(5000)
    .Interval(1000)
    .MajorGridLines(ViewBag.majorGridLines)
    .Title("Expenditure")
)
.Title("Company Revenue and Profit")
.LegendSettings(lg => lg.Visible(true))
.Tooltip(tl => tl.Enable(true)).Load("load")
.TextRender("textRender").AxisLabelRender("axisLabelRender").Render()

```

WATERFALL.CS

```

public ActionResult Index()
{
    List<WaterfallChartData> chartData = new
List<WaterfallChartData>
    {
        new WaterfallChartData { xValue = "Income", yValue = 4711 },
        new WaterfallChartData { xValue = "Sales", yValue = -1015 },
        new WaterfallChartData { xValue = "Development", yValue = -
688 },
        new WaterfallChartData { xValue = "Revenue", yValue = 1030
    },
        new WaterfallChartData { xValue = "Balance" },
        new WaterfallChartData { xValue = "Expense", yValue = -361
    },
        new WaterfallChartData { xValue = "Tax", yValue = -695 },
        new WaterfallChartData { xValue = "Net Profit" },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class WaterfallChartData
{
    public string xValue;
    public double yValue;
}

```

Series customization

The negative changes of waterfall charts is represented by using [NegativeFillColor](#) and the summary changes are represented by using [SummaryFillColor](#) properties.

By default, the negativeFillColor as '#E94649' and the summaryFillColor as '#4E81BC'.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{

```

```

        series.Width(2).
        XName("xValue").
        YName("yValue").
        IntermediateSumIndexes(ViewBag.intermediateSumIndexes).
        SumIndexes(ViewBag.sumIndexes).
        Marker(ViewBag.marker).
        NegativeFillColor("#f8b883").
        SummaryFillColor("#e56590").
        Name("USA").
        ColumnWidth(0.9).
        Type(Syncfusion.EJ2.Charts.ChartSeriesType.Waterfall).
        DataSource(ViewBag.dataSource).
        Add();
    }
).ChartArea(area => area.Border(ViewBag.ChartBorder))
.PrimaryXAxis(py =>
    py.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .MajorGridLines(ViewBag.majorGridLines)
    .PlotOffset(20)
)
.PrimaryYAxis(py =>
    py.Minimum(0)
    .Maximum(5000)
    .Interval(1000)
    .MajorGridLines(ViewBag.majorGridLines)
    .Title("Expenditure")
).Title("Company Revenue and Profit")
.LegendSettings(lg => lg.Visible(true))
.Tooltip(tl => tl.Enable(true)).Load("load")
.TextRender("textRender").AxisLabelRender("axisLabelRender").Render()

```

CUSTOM-WATERFALL.CS

```

public ActionResult Index()
{
    List<WaterfallChartData> chartData = new
List<WaterfallChartData>
    {
        new WaterfallChartData { xValue = "Income", yValue = 4711 },
        new WaterfallChartData { xValue = "Sales", yValue = -1015 },
        new WaterfallChartData { xValue = "Development", yValue = -
688 },
        new WaterfallChartData { xValue = "Revenue", yValue = 1030
    },
        new WaterfallChartData { xValue = "Balance" },
        new WaterfallChartData { xValue = "Expense", yValue = -361
    },
        new WaterfallChartData { xValue = "Tax", yValue = -695 },
        new WaterfallChartData { xValue = "Net Profit" },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class WaterfallChartData
{
    public string xValue;

```

```

        public double yValue;
    }

```

See Also

- [Data Label](#)
- [Tooltip](#)

Histogram in ASP.NET MVC Charts Component

Histogram

[Histogram Chart](#) can provide a visual display of large amounts of data that are difficult to understand in a tabular or spreadsheet form and it can be rendered by specifying the series [Type](#) to [Histogram](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Histogram).
    YName("Score").
    BinInterval(20).
    ShowNormalDistribution(true).
    ColumnWidth(0.99).
    Marker(mr => mr.Visible(true).Width(7).Height(7).DataLabel(dl =>
dl.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top).Font(ff
=> ff.FontWeight("600").Color("#ffffff")))).
    DataSource(ViewBag.ChartPoints).Add();
})
.PrimaryXAxis(px =>
    px.Maximum(100)
    .Minimum(0)
    .Title("Score of Final Examination")

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(mg=>mg.Width(0))
)
.PrimaryYAxis(px =>
    px.Minimum(0)
    .Maximum(50)
    .Interval(10)
    .Title("Number of Students")
    .LineStyle(ls=>ls.Width(0))
).Title("Examination Result")
.ChartArea(ca => ca.Border(br=>br.Width(0)))
.Tooltip(tt => tt.Enable(true)).Load("load").Render()

```

HISTOGRAM.CS

```

public IActionResult Index()
{
    List<HistogramChartData> chartData = new
List<HistogramChartData>
    {
        new HistogramChartData { Score = 5.250 },
        new HistogramChartData { Score = 7.750 },
    }
}

```

```

new HistogramChartData { Score = 0 },
new HistogramChartData { Score = 8.275 },
new HistogramChartData { Score = 9.750 },
new HistogramChartData { Score = 7.750 },
new HistogramChartData { Score = 8.275 },
new HistogramChartData { Score = 6.250 },
new HistogramChartData { Score = 5.750 },
new HistogramChartData { Score = 5.250 },
new HistogramChartData { Score = 23.000 },
new HistogramChartData { Score = 26.500 },
new HistogramChartData { Score = 27.750 },
new HistogramChartData { Score = 25.025 },
new HistogramChartData { Score = 26.500 },
new HistogramChartData { Score = 28.025 },
new HistogramChartData { Score = 29.250 },
new HistogramChartData { Score = 26.750 },
new HistogramChartData { Score = 27.250 },
new HistogramChartData { Score = 26.250 },
new HistogramChartData { Score = 25.250 },
new HistogramChartData { Score = 34.500 },
new HistogramChartData { Score = 25.625 },
new HistogramChartData { Score = 25.500 },
new HistogramChartData { Score = 26.625 },
new HistogramChartData { Score = 36.275 },
new HistogramChartData { Score = 36.250 },
new HistogramChartData { Score = 26.875 },
new HistogramChartData { Score = 40.000 },
new HistogramChartData { Score = 43.000 },
new HistogramChartData { Score = 46.500 },
new HistogramChartData { Score = 47.750 },
new HistogramChartData { Score = 45.025 },
new HistogramChartData { Score = 56.500 },
new HistogramChartData { Score = 58.025 },
new HistogramChartData { Score = 59.250 },
new HistogramChartData { Score = 56.750 },
new HistogramChartData { Score = 57.250 },
new HistogramChartData { Score = 46.250 },
new HistogramChartData { Score = 55.250 },
new HistogramChartData { Score = 44.500 },
new HistogramChartData { Score = 45.525 },
new HistogramChartData { Score = 55.500 },
new HistogramChartData { Score = 46.625 },
new HistogramChartData { Score = 46.275 },
new HistogramChartData { Score = 56.250 },
new HistogramChartData { Score = 46.875 },
new HistogramChartData { Score = 43.000 },
new HistogramChartData { Score = 46.250 },
new HistogramChartData { Score = 55.250 },
new HistogramChartData { Score = 44.500 },
new HistogramChartData { Score = 45.425 },
new HistogramChartData { Score = 55.500 },
new HistogramChartData { Score = 56.625 },
new HistogramChartData { Score = 46.275 },
new HistogramChartData { Score = 56.250 },
new HistogramChartData { Score = 46.875 },
new HistogramChartData { Score = 43.000 },
new HistogramChartData { Score = 46.250 },

```

```
new HistogramChartData { Score = 55.250},
new HistogramChartData { Score = 44.500},
new HistogramChartData { Score = 45.425},
new HistogramChartData { Score = 55.500},
new HistogramChartData { Score = 46.625},
new HistogramChartData { Score = 56.275},
new HistogramChartData { Score = 46.250},
new HistogramChartData { Score = 56.875},
new HistogramChartData { Score = 41.000},
new HistogramChartData { Score = 63.000},
new HistogramChartData { Score = 66.500},
new HistogramChartData { Score = 67.750},
new HistogramChartData { Score = 65.025},
new HistogramChartData { Score = 66.500},
new HistogramChartData { Score = 76.500},
new HistogramChartData { Score = 78.025},
new HistogramChartData { Score = 79.250},
new HistogramChartData { Score = 76.750},
new HistogramChartData { Score = 77.250},
new HistogramChartData { Score = 66.250},
new HistogramChartData { Score = 75.250},
new HistogramChartData { Score = 74.500},
new HistogramChartData { Score = 65.625},
new HistogramChartData { Score = 75.500},
new HistogramChartData { Score = 76.625},
new HistogramChartData { Score = 76.275},
new HistogramChartData { Score = 66.250},
new HistogramChartData { Score = 66.875},
new HistogramChartData { Score = 80.000},
new HistogramChartData { Score = 85.250},
new HistogramChartData { Score = 87.750},
new HistogramChartData { Score = 89.000},
new HistogramChartData { Score = 88.275},
new HistogramChartData { Score = 89.750},
new HistogramChartData { Score = 97.750},
new HistogramChartData { Score = 98.275},
new HistogramChartData { Score = 96.250},
new HistogramChartData { Score = 95.750},
new HistogramChartData { Score = 95.250}
};
ViewBag.dataSource = chartData;
return View();
}
public class HistogramChartData
{
    public double Score;
}
```

See Also

- [Data Label](#)
- [Tooltip](#)

Error Bar in ASP.NET MVC Charts Component

Error Bar

Error bars are graphical representations of the variability of data and used on graphs to indicate the error or uncertainty in a reported measurement. To render the error bar for the series, set [Visible](#) as `true` in error bar object.

CSHTML

```
@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .Name("Sales")
        .XName("x")
        .YName("y")
        .Marker(ViewBag.marker)
        .ErrorBar(ViewBag.errorBar)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.MinorGridLines(ViewBag.gridlines)
        .Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}%")
        .LineStyle(ViewBag.linestyle)
        .Minimum(15)
        .Maximum(45)
    )
    .Title("Sales Distribution of Car by Region")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Render()
)
```

ERRORBAR.CS

```
public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2005, y= 28 },
        new ErrorBarData{ x= 2006, y= 25 },
        new ErrorBarData{ x= 2007, y= 26 },
        new ErrorBarData{ x= 2008, y= 27 },
        new ErrorBarData{ x= 2009, y= 32 },
        new ErrorBarData{ x= 2010, y= 35 },
        new ErrorBarData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ErrorBarData
{
}
```

```

        public double x;
        public double y;
    }

```

Error Bar Type

To change the error bar rendering type using [Type](#) option of error bar. To change the error bar line length you can use [VerticalError](#) property.

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .Name("Sales")
        .XName("x")
        .YName("y")
        .Marker(ViewBag.marker)
        .ErrorBar(ViewBag.errorBar)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.MinorGridLines(ViewBag.gridlines)
        .Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}%")
        .LineStyle(ViewBag.linestyle)
        .Minimum(15)
        .Maximum(45)
    )
    .Title("Sales Distribution of Car by Region")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Render()
)

```

ERROR-TYPE.CS

```

public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2005, y= 28 },
        new ErrorBarData{ x= 2006, y= 25 },
        new ErrorBarData{ x= 2007, y= 26 },
        new ErrorBarData{ x= 2008, y= 27 },
        new ErrorBarData{ x= 2009, y= 32 },
        new ErrorBarData{ x= 2010, y= 35 },
        new ErrorBarData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

```



```

    }
    public class ErrorBarData
    {
        public double x;
        public double y;
    }

```

Customizing Error Bar Type

To customize the error bar type, set error bar [Type](#) as **Custom** and then change the horizontal/vertical positive and negative error of error bar.

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr => {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .Name("Sales")
        .XName("x")
        .YName("y")
        .ErrorBar(eb =>
            eb.Type(Syncfusion.EJ2.Charts.ErrorBarType.Custom).Visible(true).Mode(Syncfusion.EJ2.Charts.ErrorBarMode.Both)
                .VerticalPositiveError(2).HorizontalPositiveError(1).VerticalNegativeError(2)
                .HorizontalNegativeError(1))
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}%")
        .Minimum(15)
        .Maximum(45)
    )
    .Title("Sales Distribution of Car by Region")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Render()
)

```

CUSTOM-ERROR.CS

```

public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2005, y= 28 },
        new ErrorBarData{ x= 2006, y= 25 },
        new ErrorBarData{ x= 2007, y= 26 },
        new ErrorBarData{ x= 2008, y= 27 },
        new ErrorBarData{ x= 2009, y= 32 },
        new ErrorBarData{ x= 2010, y= 35 },
        new ErrorBarData{ x= 2011, y= 30 }
    };
}

```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class ErrorBarData
    {
        public double x;
        public double y;
    }

```

Error Bar Mode

Error bar mode is used to define whether the error bar line has to be drawn horizontally, vertically or in both side. To change the error bar mode use [Mode](#) option.

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .Name("Sales")
        .XName("x")
        .YName("y")
        .Marker(ViewBag.marker)
        .ErrorBar(ViewBag.errorBar)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.MinorGridLines(ViewBag.gridlines)
        .Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}%")
        .LineStyle(ViewBag.linestyle)
        .Minimum(15)
        .Maximum(45)
    )
    .Title("Sales Distribution of Car by Region")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Render()
)

```

ERROR-MODE.CS

```

public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2005, y= 28 },
        new ErrorBarData{ x= 2006, y= 25 },
        new ErrorBarData{ x= 2007, y= 26 },
        new ErrorBarData{ x= 2008, y= 27 },
        new ErrorBarData{ x= 2009, y= 32 },
    }
}

```

```

        new ErrorBarData{ x= 2010, y= 35 },
        new ErrorBarData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ErrorBarData
{
    public double x;
    public double y;
}

```

Error Bar Direction

To change the error bar direction to plus, minus or both side using [Direction](#) option.

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
            .Name("Sales")
            .XName("x")
            .YName("y")
            .Marker(ViewBag.marker)
            .ErrorBar(ViewBag.errorBar)
            .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.MinorGridLines(ViewBag.gridlines)
            .Interval(1)
            .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}%")
            .LineStyle(ViewBag.linestyle)
            .Minimum(15)
            .Maximum(45)
    )
    .Title("Sales Distribution of Car by Region")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Render()
)

```

ERROR-DIRECTION.CS

```

public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2005, y= 28 },
        new ErrorBarData{ x= 2006, y= 25 },
        new ErrorBarData{ x= 2007, y= 26 },
        new ErrorBarData{ x= 2008, y= 27 },
    }
}

```

```

        new ErrorBarData{ x= 2009, y= 32 },
        new ErrorBarData{ x= 2010, y= 35 },
        new ErrorBarData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ErrorBarData
{
    public double x;
    public double y;
}

```

Customizing Error Bar Cap

To customize the error bar cap length, width and fill color, you can use [ErrorBarCap](#) option.

CSHTML

```

@ (Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .Name("Sales")
        .XName("x")
        .YName("y")
        .Marker(ViewBag.marker)
        .ErrorBar(ViewBag.errorBar)
        .DataSource(ViewBag.dataSource).Add();
    })
    .PrimaryXAxis(xaxis =>
        xaxis.MinorGridLines(ViewBag.gridlines)
        .Interval(1)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(yaxis =>
        yaxis.LabelFormat("{value}%")
        .LineStyle(ViewBag.linestyle)
        .Minimum(15)
        .Maximum(45)
    )
    .Title("Sales Distribution of Car by Region")
    .Tooltip(tp => tp.Enable(true))
    .LegendSettings(lg => lg.Visible(false))
    .Load("load").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Render()
)

```

ERROR-CAP.CS

```

public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2005, y= 28 },
        new ErrorBarData{ x= 2006, y= 25 },
        new ErrorBarData{ x= 2007, y= 26 },
    }
}

```

```

        new ErrorBarData{ x= 2008, y= 27 },
        new ErrorBarData{ x= 2009, y= 32 },
        new ErrorBarData{ x= 2010, y= 35 },
        new ErrorBarData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ErrorBarData
{
    public double x;
    public double y;
}

```

Customizing Error Bar Color

To customize the error bar color for individual errors, use the [ErrorBarColorMapping](#) property. You can also customize the vertical error, horizontal error, horizontal negative and positive error and vertical negative and positive error for an individual point using [VerticalError](#), [HorizontalError](#), [HorizontalNegativeError](#), [HorizontalPositiveError](#), [VerticalNegativeError](#) and [VerticalPositiveError](#) properties.

CSHTML

```

@(Html.EJS().Chart("container")
    .Series(sr =>
    {
        sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .XName("x")
        .YName("y")
        .Name("India")
        .Width(2)
        .Marker(mr=>mr.Visible(true))
        .ErrorBar(eb =>
        eb.VerticalError("error").Visible(true).ErrorBarColorMapping("color"))
        .DataSource(ViewBag.dataSource).Add();
    }).PrimaryXAxis(px =>
        px.Minimum(2005)
        .Maximum(2012)
        .Interval(1)
        .Title("Year")
    )
    .Title("Unemployment rate (%)").Load("load").Render()
)

```

ERROR-BAR-COLOR.CS

```

public ActionResult Index()
{
    List<ErrorBarData> chartData = new List<ErrorBarData>
    {
        new ErrorBarData{ x= 2006, y= 7.8, color= 'red', error= 1.5
    },
        new ErrorBarData{ x= 2007, y= 7.2, color= 'green', error= 2.5
    },
    },
}

```

```

        new ErrorBarData{ x= 2008, y= 6.8, color= 'blue', error= 3.5
    },
        new ErrorBarData{ x= 2009, y= 5.7, color= 'yellow', error=
1.5 },
        new ErrorBarData{ x= 2010, y= 10.8, color= 'grey', error= 0.5
    },
        new ErrorBarData{ x= 2011, y= 9.8, color= 'brown', error= 1 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ErrorBarData
{
    public double x;
    public double y;
    public string color;
    public double error;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Vertical Chart in ASP.NET MVC Charts Component

In EJ2 chart, you can draw a chart in vertical manner by changing orientation of the axis. All series types support this feature. You can use `IsTransposed` property in chart to render a chart in vertical manner.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).
    Marker(ViewBag.Marker).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).
    Add();
}).
PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
).
Title("Olympic Medal Counts - RIO")
.IsTransposed(true).Render()

```

VERTICAL.CS

```

public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {

```

```

        new VerticalData{ x= 2005, y= 28 },
        new VerticalData{ x= 2006, y= 25 },
        new VerticalData{ x= 2007, y= 26 },
        new VerticalData{ x= 2008, y= 27 },
        new VerticalData{ x= 2009, y= 32 },
        new VerticalData{ x= 2010, y= 35 },
        new VerticalData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class VerticalData
{
    public double x;
    public double y;
}

```

See Also

- [Data Label](#)
- [Tooltip](#)

Pareto in ASP.NET MVC Charts Component

Pareto

Pareto charts are used to find the cumulative values of data in different categories. It is a combination of Column and Line series. The initial values are represented by column chart and the cumulative values are represented by Line chart. To render a pareto chart, use series [Type](#) as [Pareto](#).

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Pareto).
    Marker(mr => mr.Visible(true).Height(10).Width(10)).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).
    Add();
})).
PrimaryXAxis(px =>
    px.Title("Defects")
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
).
PrimaryYAxis(py =>
    py.Title("Frequency")
    .Minimum(0)
    .Maximum(150)
    .Interval(30)
)
.Render()
)

```

PARETO.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ x= "Traffic", y=56 },
        new ColumnChartData{ x="Child care", y=44.8 },
        new ColumnChartData{ x= "Transport", y=27.2 },
        new ColumnChartData{ x= "Weather", y=19.6},
        new ColumnChartData{ x= "Emergency", y=6.6 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double y;
};

```

Pareto customization

The pareto line series can be customized by using the [Marker](#), [Width](#), [DashArray](#), and [Fill](#) properties in the [ParetoOptions](#). The secondary axis for the pareto series can be shown or hidden using the [ShowAxis](#) property.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.XName("DefectCategory")
    .YName("Y")
    .Name("Defect")
    .Width(2)
    .Opacity(0.75)
    .ColumnWidth(0.4)
    .ParetoOptions(pr=>pr.DashArray("3,2").Marker(mr =>
mr.Visible(true).Width(7).Height(7).IsFilled(true)).Width(2).Fill("#F7523F")
)
    .DataSource(ViewBag.ChartPoints)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Pareto).Add();
})).
PrimaryXAxis(px =>
    px.Interval(1)

.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate45)
    .MajorGridLines(mg => mg.Width(0))
    .MinorGridLines(mg => mg.Width(0))
    .MajorTickLines(mt => mt.Width(0))
    .MinorTickLines(mt => mt.Width(0))
    .LineStyle(ls => ls.Width(0))
).
PrimaryYAxis(py =>
    py.Title("Frequency of Occurrence")
    .Minimum(0)

```



```

        .Maximum(25)
        .Interval(5)
        .MajorGridLines(mg => mg.Width(1))
        .MinorGridLines(mg => mg.Width(1))
        .MajorTickLines(mt => mt.Width(0))
        .MinorTickLines(mt => mt.Width(0))
        .LineStyle(ls => ls.Width(0))
    )
    .Title("Defects in Shirts")
    .LegendSettings(leg => leg.Visible(true).EnableHighlight(true))
    .Tooltip(tp => tp.Enable(true).Shared(true).Format("${series.name} :
<b>${point.y}</b>"))
    .ChartArea(area => area.Border(br => br.Width(0)))
    .Render()
)

```

PARETO-CUSTOMIZATION.CS

```

public ActionResult Index()
{
    List<ParetoChartData> ChartPoints = new List<ParetoChartData>
    {
        new ParetoChartData { DefectCategory = "Button Defect", Y =
23 },
        new ParetoChartData { DefectCategory = "Pocket Defect", Y =
16 },
        new ParetoChartData { DefectCategory = "Collar Defect", Y =
10 },
        new ParetoChartData { DefectCategory = "Cuff Defect", Y = 7
},
        new ParetoChartData { DefectCategory = "Sleeve Defect", Y =
6 },
        new ParetoChartData { DefectCategory = "Other Defect", Y = 2
    }
    };
    ViewBag.ChartPoints = ChartPoints;
    return View();
}

public class ParetoChartData
{
    public string DefectCategory;
    public double Y;
}

```

See also

- [Data Label](#)
- [Tooltip](#)

Chart Series in Chart Component

Multiple Series

You can add multiple series to the chart by using [Series](#) property. The series are rendered in the order as it is added to the series array.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    .Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource1).
    Name("Silver").
    .Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.IsIndexed(true)).
Title("Olympic Medal Counts - RIO").Render()
```

MULTIPLE-SERIES.CS

```
public IActionResult Column()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}
```

Combination Series

Combination of different types like Line, column etc, can be rendered in a chart.

Note: Bar series cannot be combined with any other series as the axis orientation is different from other series.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    .Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource1).
    Name("Silver").
    .Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.IsIndexed(true)).
Title("Olympic Medal Counts - RIO").Render()
```

COMBINATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Charts;
namespace EJ2CoreSampleBrowser.Controllers.Chart
{
    public partial class ChartController : Controller
    {
        public IActionResult Column()
        {
            List<CombinationSeriesData> chartData = new
            List<CombinationSeriesData>
            {
                new CombinationSeriesData { x= "2007", y= 1 },
                new CombinationSeriesData { x= "2008", y= 0.25 },
                new CombinationSeriesData { x= "2009", y= 0.1 },
                new CombinationSeriesData { x= "2010", y= 1 },
                new CombinationSeriesData { x= "2011", y= 0.1 },
                new CombinationSeriesData { x= "2012", y= -0.25 },
                new CombinationSeriesData { x= "2013", y= 0.25 },
                new CombinationSeriesData { x= "2014", y= 0.6 }
            };
            ViewBag.dataSource = chartData;
        }
    }
}
```

```

        List<CombinationSeriesData> chartData1 = new
List<CombinationSeriesData>
    {
        new CombinationSeriesData { x= "2007", y= 0.5 },
        new CombinationSeriesData { x= "2008", y= 0.35 },
        new CombinationSeriesData { x= "2009", y= 0.9 },
        new CombinationSeriesData { x= "2010", y= 0.5 },
        new CombinationSeriesData { x= "2011", y= 0.25 },
        new CombinationSeriesData { x= "2012", y= -0.5 },
        new CombinationSeriesData { x= "2013", y= 0.5 },
        new CombinationSeriesData { x= "2014", y= 0.6 }
    };
    ViewBag.dataSource1 = chartData1;
    List<CombinationSeriesData> chartData2 = new
List<CombinationSeriesData>
    {
        new CombinationSeriesData { x= "2007", y= 1.5 },
        new CombinationSeriesData { x= "2008", y= 0.35 },
        new CombinationSeriesData { x= "2009", y= -2.7 },
        new CombinationSeriesData { x= "2010", y= 0.5 },
        new CombinationSeriesData { x= "2011", y= 0.25 },
        new CombinationSeriesData { x= "2012", y= -0.1 },
        new CombinationSeriesData { x= "2013", y= -0.3 },
        new CombinationSeriesData { x= "2014", y= -0.6 }
    };
    ViewBag.dataSource2 = chartData2;
    List<CombinationSeriesData> chartData3 = new
List<CombinationSeriesData>
    {
        new CombinationSeriesData { x= "2007", y= -1 },
        new CombinationSeriesData { x= "2008", y= -.35 },
        new CombinationSeriesData { x= "2009", y= -0.3 },
        new CombinationSeriesData { x= "2010", y= -0.5 },
        new CombinationSeriesData { x= "2011", y= 1 },
        new CombinationSeriesData { x= "2012", y= -0.4 },
        new CombinationSeriesData { x= "2013", y= 0.1 },
        new CombinationSeriesData { x= "2014", y= -0.6 }
    };
    ViewBag.dataSource3 = chartData3;
    List<CombinationSeriesData> chartData4 = new
List<CombinationSeriesData>
    {
        new CombinationSeriesData { x= "2007", y= 2 },
        new CombinationSeriesData { x= "2008", y= 0.1 },
        new CombinationSeriesData { x= "2009", y= -2.7 },
        new CombinationSeriesData { x= "2010", y= 1.8 },
        new CombinationSeriesData { x= "2011", y= 2 },
        new CombinationSeriesData { x= "2012", y= 0.4 },
        new CombinationSeriesData { x= "2013", y= 0.9 },
        new CombinationSeriesData { x= "2014", y= 0.4 }
    };
    ViewBag.dataSource4 = chartData4;
    return View();
}
public class CombinationSeriesData
{
    public string x;

```

```

        public double y;
    }
}

```

Enable Complex Property in Series

By setting [EnableComplexProperty](#) value as **true** in series you can bind complex data to the chart.

CSHTML

```

@Html.EJS().Chart("container").PrimaryXAxis(px => px.Title("data").
ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Series(series =>
{
    series.Type(
        Syncfusion.EJ2.Charts.ChartSeriesType.Column
    ).Name("Product X").EnableComplexProperty(true).Add();
    series.Type(
        Syncfusion.EJ2.Charts.ChartSeriesType.Column
    ).Name("Product Y").EnableComplexProperty(true).Add();
}).PrimaryYAxis(py => py.Title("Profit
($)")).Load("load").Title("Sales History of Product X").Render()
<script>

    var load = function (args) {
        var data = [
            { group: { x: 'Aaa', y: 10 }, y: 20 },
            { group: { x: 'Baa', y: 30 }, y: 10 }
        ];
        args.chart.series[0].dataSource = data;
        args.chart.series[1].dataSource = data;
        args.chart.series[0].xName = "group.x";
        args.chart.series[0].yName = "group.y";
        args.chart.series[1].xName = "group.x";
        args.chart.series[1].yName = "y";
    }
</script>

```

COMPLEX.CS

```

public ActionResult Index()
{
    return View();
}

```

<!-- markdownlint-disable MD036 -->

Technical Indicators

A [Technical indicator](#) is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

Chart supports 10 types of technical indicators.

Accumulation Distribution

Accumulation Distribution combines price and volume to show how money may be flowing into or out of stock. To render a [Accumulation Distribution Indicator](#), use indicator [Type](#) as AccumulationDistribution. To calculate the signal line [Volume](#) field is additionally added with dataSource.

CSHTML

```
<ejs-chart id="ADContainer" title="AAPL - 2012-2017"
axisLabelRender="axisLabel" load="window.onChartLoad">
    <e-chart-primaryxaxis valueType="DateTime" intervalType="Months"
zoomFactor=0.2 zoomPosition=0.6></e-chart-primaryxaxis>
    <e-chart-tooltipssettings enable="true" shared="true"></e-chart-
tooltipssettings>
    <e-chart-legendsettings visible="false"></e-chart-legendsettings>
    <e-chart-zoomsettings enableSelectionZooming="true" mode="X"></e-
chart-zoomsettings>
    <e-chart-crosshairsettings enable=true lineType="Vertical"></e-
chart-crosshairsettings>
    <e-chart-axes>
        <e-chart-axis name="secondary"
                        opposedPosition=true rowIndex=0
interval=60000000000
                        minimum=-7000000000 maximum=5000000000
title="Accumulation Distribution">
            </e-chart-axis>
        </e-chart-axes>
        <e-series-collection>
            <e-series xName="x" yName="y" low="low" high="high"
close="close" volume="volume" open="open" width=2
                        name="Apple Inc" bearFillColor="#2ecd71"
bullFillColor="#e74c3d"
type="@Syncfusion.EJ2.Charts.ChartSeriesType.Candle"></e-series>
            </e-series-collection>
            <e-indicators>
                <e-indicator
type="@Syncfusion.EJ2.Charts.TechnicalIndicators.AccumulationDistribution"
field="Close" seriesName="Apple Inc" yAxisName="secondary" fill="#6063ff"
period=3>
                </e-indicator>
            </e-indicators>
        </ejs-chart>
<script src="chart/technical-indicators/ad/financial-data.js"></script>
<script>
    window.onChartLoad = function (args) {
        args.chart.series[0].dataSource = window.chartData;
    }
</script>
```

AD.CS

```
public IActionResult Index()
{
    return View();
}
```

Average True Range (ATR)

ATR measures the stock volatility by comparing the current value with the previous value. To render a Average True Range (ATR) Indicator, use indicator [Type](#) as `Atr`.

CSHTML

```
<script src="chart/technical-indicators/atr/financial-data.js"></script>
@Html.EJS().Chart("container-vertical").Rows(rows =>
{
    rows.Height("40%").Add();
    rows.Height("60%").Add();
}).Axes(ax =>
{
    ax.Name("secondary").LineStyle(ViewBag.Line).Title("ATR").MajorTickLines(ViewBag.majorGridLines).OpposedPosition(true).Minimum(0).Maximum(14).Interval(7).RowIndex(0).Add();
}
).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("y").High("high").Low("low").Open("open").Close("close").Volume("volume").BearFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name("Apple Inc").Width(2).Add();
}).Indicators(id =>
{
    id.Fill("#6063ff").KPeriod(2).DPeriod(3).YAxisName("secondary").Period(3).Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Atr).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple Inc").Add();
}).ZoomSettings(zn =>
zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X).Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
).PrimaryXAxis(px =>
px.ZoomFactor(0.2).ZoomPosition(0.6).MajorGridLines(ViewBag.Line).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
).PrimaryYAxis(py =>
py.LineStyle(ViewBag.Line).RowIndex(1).OpposedPosition(true).Title("Price").Minimum(80).Maximum(170).Interval(30).LabelFormat("{value}M").PlotOffset(25)
).Title("AAPL 2012-2017").ChartArea(area =>
area.Border(ViewBag.ChartBorder).Tooltip(tl =>
tl.Enable(true).Shared(true)).LegendSettings(lg =>
lg.Visible(false)).Load("load").Render()
<script>
    var dataSource = window.chartData;
</script>
```

ATR.CS

```
public IActionResult Index()
{
    return View();
}
```

}

Bollinger Band

A chart overlay that shows the upper and lower limits of normal price movements based on the standard deviation of prices. To render a Bollinger Band, use indicator [Type](#) as `BollingerBand`. Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine). Bollinger Band default values of the [Period](#) is 14 and [StandardDeviations](#) is 2.

CSHTML

```
<script src="chart/technical-indicators/bollinger/financial-
data.js"></script>
@ (Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle)
    .XName("x")
    .YName("y")
    .High("high")
    .Low("low")
    .Open("open")
    .Close("close")
    .Volume("volume")
    .BearFillColor("#2ecd71")
    .BullFillColor("#e74c3d")
    .DataSource("dataSource")
    .Name("Apple Inc")
    .Width(2).Add();
}).Indicators(id =>
{
    id.Fill("#6063ff")
    .Period(14)
    .UpperLine(ViewBag.upperline)
    .LowerLine(ViewBag.lowerline)
    .Animation(ViewBag.animation)
    .Type(Syncfusion.EJ2.Charts.TechnicalIndicators.BollingerBands)
    .Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close)
    .SeriesName("Apple Inc").Add();
})
.ZoomSettings(zn => zn.EnableSelectionZooming(true)
    .Mode(Syncfusion.EJ2.Charts.ZoomMode.X))
.Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true))
.PrimaryXAxis(px =>
px.ZoomFactor(0.2).CrosshairTooltip(ViewBag.crosshairtooltip).MajorGridLines
(ViewBag.gridlines).ZoomPosition(0.6).ValueType(Syncfusion.EJ2.Charts.ValueT
ype.DateTime))
.PrimaryYAxis(py =>
py.Title("Price").Minimum(50).Maximum(170).Interval(30).LineStyle(ViewBag.li
nestyle).LabelFormat("{value}M"))
.Title("AAPL 2012-2017")
.Tooltip(tl => tl.Enable(true).Shared(true))
.LegendSettings(lg => lg.Visible(false))
.ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Load("load").Render()
)
```



```
<script>
    var dataSource = window.chartData;
</script>
```

BOLLINGER.CS

```
public IActionResult Index()
{
    return View();
}
```

Customization of BollingerBand

stroke, stroke-width, and color of upperLine can be customized by using [UpperLine](#), and the lowerLine can be customized by using [LowerLine](#) properties of indicator.

CSHTML

```
@(Html.EJS().Chart("container").EnableAnimation(true).Crosshair(ch =>
ch.Enable(true).LineType(Syncfusion.EJ2.Charts.LineType.Vertical)).ChartArea
(ca => ca.Border(br => br.Width(0))).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).
        XName("x").
        YName("y").
        Low("low").
        High("high").
        Close("close").
        Volume("volume").
        Open("open").
        Name("Apple Inc").
        BearFillColor("#2ecd71").
        BullFillColor("#e74c3d").
        DataSource(ViewBag.dataSource).Width(2).Add();
}).
    Indicators(ind => {
        ind.Type(Syncfusion.EJ2.Charts.TechnicalIndicators.BollingerBands).
            Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).
            SeriesName("Apple Inc").
            Fill("#606eff").
            Period(14).
            Animation(an => an.Enable(true)).
            UpperLine(ul => ul.Color("#ffb735").Width(1)).
            LowerLine(ll => ll.Color("#f2ec2f").Width(1)).Add();
    }).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).CrosshairTooltip(cr
=>
cr.Enable(true)).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Months).MajorGridLines(mg => mg.Width(0))).
    PrimaryYAxis(py =>
py.Title("Price").LabelFormat("${value}").Minimum(0).Maximum(180).Interval(30).LineStyle(ls => ls.Width(0))).
    Tooltip(tp => tp.Enable(true).Shared(true)).
    Render()
)
```

CUSTOM-BOLLINGER.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data { x= new DateTime(2012,04,02), open= 85.9757, high=
90.6657, low= 85.7685, close= 90.5257, volume= 660187068 },
        new Data { x= new DateTime(2012,04,09), open= 89.4471, high=
92, low= 86.2157, close= 86.4614, volume= 912634864 },
        new Data { x= new DateTime(2012,04,16), open= 87.1514, high=
88.6071, low= 81.4885, close= 81.8543, volume= 1221746066 },
        new Data { x= new DateTime(2012,04,23), open= 81.5157, high=
88.2857, low= 79.2857, close= 86.1428, volume= 965935749 },
        new Data { x= new DateTime(2012,04,30), open= 85.4, high=
85.4857, low= 80.7385, close= 80.75, volume= 615249365 },
        new Data { x= new DateTime(2012,05,07), open= 80.2143, high=
82.2685, low= 79.8185, close= 80.9585, volume= 541742692 },
        new Data { x= new DateTime(2012,05,14), open= 80.3671, high=
81.0728, low= 74.5971, close= 75.7685, volume= 708126233 },
        new Data { x= new DateTime(2012,05,21), open= 76.3571, high=
82.3571, low= 76.2928, close= 80.3271, volume= 682076215 },
        new Data { x= new DateTime(2012,05,28), open= 81.5571, high=
83.0714, low= 80.0743, close= 80.1414, volume= 480059584 },
        new Data { x= new DateTime(2012,06,04), open= 80.2143, high=
82.9405, low= 78.3571, close= 82.9028, volume= 517577005 },
        new Data { x= new DateTime(2012,06,11), open= 83.96, high=
84.0714, low= 80.9571, close= 82.0185, volume= 499693120 },
        new Data { x= new DateTime(2012,06,18), open= 81.5657, high=
84.2857, low= 81.4814, close= 83.1571, volume= 442172142 },
        new Data { x= new DateTime(2012,06,25), open= 82.4714, high=
83.4285, low= 80.8014, close= 83.4285, volume= 371529102 },
        new Data { x= new DateTime(2012,07,02), open= 83.5328, high=
87.7628, low= 83.3714, close= 86.5543, volume= 385906790 },
        new Data { x= new DateTime(2012,07,09), open= 86.4714, high=
88.5528, low= 84.6685, close= 86.4243, volume= 524235196 },
        new Data { x= new DateTime(2012,07,16), open= 86.4457, high=
87.9071, low= 86.1643, close= 86.3285, volume= 419537217 },
        new Data { x= new DateTime(2012,07,23), open= 84.9143, high=
87.0971, low= 81.4285, close= 83.5943, volume= 680773023 },
        new Data { x= new DateTime(2012,07,30), open= 84.4171, high=
88.2828, low= 83.9743, close= 87.9571, volume= 475109323 },
        new Data { x= new DateTime(2012,08,06), open= 88.1843, high=
89.2857, low= 87.8943, close= 88.8143, volume= 312826308 },
        new Data { x= new DateTime(2012,08,13), open= 89.0557, high=
92.5985, low= 89.0357, close= 92.5871, volume= 392867193 },
        new Data { x= new DateTime(2012,08,20), open= 92.8585, high=
96.4114, low= 92.5871, close= 94.746, volume= 708614692 },
        new Data { x= new DateTime(2012,08,27), open= 97.1414, high=
97.2671, low= 93.8928, close= 95.0343, volume= 383807217 },
        new Data { x= new DateTime(2012,09,03), open= 95.1085, high=
97.4971, low= 94.9285, close= 97.2057, volume= 355722047 },
        new Data { x= new DateTime(2012,09,10), open= 97.2071, high=
99.5685, low= 93.7143, close= 98.7543, volume= 724042207 },
    }
}

```

```

        new Data { x= new DateTime(2012,09,17), open= 99.9071, high=
100.7243, low= 99.0885, close= 100.0135, volume= 500166040 },
        new Data { x= new DateTime(2012,09,24), open= 98.1228, high=
99.3028, low= 94.3357, close= 95.3007, volume= 714507994 },
        new Data { x= new DateTime(2012,10,01), open= 95.88, high=
96.6785, low= 92.95, close= 93.2271, volume= 638543622 },
        new Data { x= new DateTime(2012,10,08), open= 92.4114, high=
92.5085, low= 89.0785, close= 89.9591, volume= 747127724 },
        new Data { x= new DateTime(2012,10,15), open= 90.3357, high=
93.2557, low= 87.0885, close= 87.12, volume= 646996264 },
        new Data { x= new DateTime(2012,10,22), open= 87.4885, high=
90.7685, low= 84.4285, close= 86.2857, volume= 866040680 },
        new Data { x= new DateTime(2012,10,29), open= 84.9828, high=
86.1428, low= 82.1071, close= 82.4, volume= 367371310 },
        new Data { x= new DateTime(2012,11,05), open= 83.3593, high=
84.3914, low= 76.2457, close= 78.1514, volume= 919719846 },
        new Data { x= new DateTime(2012,11,12), open= 79.1643, high=
79.2143, low= 72.25, close= 75.3825, volume= 894382149 },
        new Data { x= new DateTime(2012,11,19), open= 77.2443, high=
81.7143, low= 77.1257, close= 81.6428, volume= 527416747 },
        new Data { x= new DateTime(2012,11,26), open= 82.2714, high=
84.8928, low= 81.7514, close= 83.6114, volume= 646467974 },
        new Data { x= new DateTime(2012,12,03), open= 84.8071, high=
84.9414, low= 74.09, close= 76.1785, volume= 980096264 },
        new Data { x= new DateTime(2012,12,10), open= 75, high=
78.5085, low= 72.2257, close= 72.8277, volume= 835016110 },
        new Data { x= new DateTime(2012,12,17), open= 72.7043, high=
76.4143, low= 71.6043, close= 74.19, volume= 726150329 },
        new Data { x= new DateTime(2012,12,24), open= 74.3357, high=
74.8928, low= 72.0943, close= 72.7984, volume= 321104733 },
        new Data { x= new DateTime(2012,12,31), open= 72.9328, high=
79.2857, low= 72.7143, close= 75.2857, volume= 540854882 },
        new Data { x= new DateTime(2013,01,07), open= 74.5714, high=
75.9843, low= 73.6, close= 74.3285, volume= 574594262 },
        new Data { x= new DateTime(2013,01,14), open= 71.8114, high=
72.9643, low= 69.0543, close= 71.4285, volume= 803105621 },
        new Data { x= new DateTime(2013,01,21), open= 72.08, high=
73.57, low= 62.1428, close= 62.84, volume= 971912560 },
        new Data { x= new DateTime(2013,01,28), open= 62.5464, high=
66.0857, low= 62.2657, close= 64.8028, volume= 656549587 },
        new Data { x= new DateTime(2013,02,04), open= 64.8443, high=
68.4014, low= 63.1428, close= 67.8543, volume= 743778993 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public DateTime x;
    public double open;
    public double high;
    public double low;
    public double close;
    public double volume;
}

```

Exponential Moving Average (EMA)

Moving average Indicators are used to define the direction of the trend. To render a EMA Indicator, use indicator [Type](#) as [Ema](#).

CSHTML

```
<script src="chart/technical-indicators/ema/financial-data.js"></script>
@ (Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle)
    .XName("x")
    .YName("y")
    .High("high")
    .Low("low")
    .Open("open")
    .Close("close")
    .Volume("volume")
    .BearFillColor("#2ecd71")
    .BullFillColor("#e74c3d")
    .DataSource("dataSource")
    .Name("Apple Inc")
    .Width(2).Add();
}).Indicators(id =>
{
    id.Fill("#6063ff")
    .Period(14)
    .Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Ema)
    .Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close)
    .SeriesName("Apple Inc").Add();
})
.ZoomSettings(zn => zn.EnableSelectionZooming(true)
    .Mode(Syncfusion.EJ2.Charts.ZoomMode.X))
.Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true))
    .PrimaryXAxis(px =>
px.ZoomFactor(0.2).ZoomPosition(0.6).MajorGridLines(ViewBag.border).ValueType
(Syncfusion.EJ2.Charts.ValueType.DateTime))
    .PrimaryYAxis(py =>
py.Title("Price").Minimum(50).LineStyle(ViewBag.border).Maximum(170).Interval(30).LabelFormat("{value}M"))
    .Title("AAPL 2012-2017")
    .Tooltip(tl => tl.Enable(true).Shared(true))
    .LegendSettings(lg => lg.Visible(false))
    .ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Load("load").Render()
)
</script>
var dataSource = window.chartData;
</script>
```

EMA.CS

```
public IActionResult Index()
{
    return View();
}
```

Momentum

Momentum shows the speed at which the price of the stock is changing. To render a Momentum indicator, use indicator [Type](#) as **Momentum**. Momentum indicator will be represented by two lines (upperLine, signalLine). In momentum indicator the upperBand value is always render at the value 100.

CSHTML

```
<script src="chart/technical-indicators/momentum/financial-
data.js"></script>
    @Html.EJS().Chart("container-vertical").Rows(rows =>
    {
        rows.Height("40%").Add();
        rows.Height("60%").Add();
    }).Axes(ax =>
    {

ax.Name("secondary").OpposedPosition(true).Minimum(80).Maximum(120).Interval
(20).Title("Momentum").LineStyle(ViewBag.lineStyle).MajorGridLines(ViewBag.m
ajorGridLines).MajorTickLines(ViewBag.majorTickLines).RowIndex(0).Add();
    }
    ).Series(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("
y").High("high").Low("low").Open("open").Close("close").Volume("volume").Bea
rFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name
("Apple Inc").Width(2).Add();
    }).Indicators(id =>
    {
        id.Fill("#6063ff")
        .YAxisName("secondary")
        .UpperLine(ViewBag.upperline)
        .Period(3)
        .Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Momentum)
        .Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close)
        .SeriesName("Apple Inc").Add();
    })
    .ZoomSettings(zn => zn.EnableSelectionZooming(true)
    .Mode(Syncfusion.EJ2.Charts.ZoomMode.X))
    .Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
    )
    .PrimaryXAxis(px => px.ZoomFactor(0.2)
    .ZoomPosition(0.6)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(ViewBag.majorGridLines)
    .CrosshairTooltip(ViewBag.crosshairTooltip)
    ).PrimaryYAxis(py => py.RowIndex(1)
    .OpposedPosition(true).Title("Price")
    .Minimum(50).Maximum(170)
    .Interval(30)
    .LabelFormat("{value}M")
    .PlotOffset(25)
    .LineStyle(ViewBag.lineStyle)
    ).Title("AAPL 2012-2017")
```

```

        .ChartArea(area => area.Border(ViewBag.ChartBorder))
        .Tooltip(tl => tl.Enable(true).Shared(true))
        .LegendSettings(lg => lg.Visible(false)).Load("load").Render()
</script>
    var dataSource = window.chartData;
</script>

```

MOMENTUM.CS

```

public IActionResult Index()
{
    return View();
}

```

Customization of MomentumIndicator

stroke, stroke-width, and color of upperLine can be customized by using, [UpperLine](#), property of indicator.

CSHTML

```

<script src="chart/technical-indicators/momentum/financial-
data.js"></script>
    @Html.EJS().Chart("container-vertical").Rows(rows =>
    {
        rows.Height("40%").Add();
        rows.Height("60%").Add();
    }).Axes(ax =>
    {
        ax.Name("secondary").OpposedPosition(true).Minimum(80).Maximum(120).Interval
(20).Title("Momentum").LineStyle(ViewBag.lineStyle).MajorGridLines(ViewBag.m
ajorGridLines).MajorTickLines(ViewBag.majorTickLines).RowIndex(0).Add();
    }
    ).Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("
y").High("high").Low("low").Open("open").Close("close").Volume("volume").Bea
rFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name
("Apple Inc").Width(2).Add();
    }).Indicators(id =>
    {
        id.Fill("#6063ff")
        .YAxisName("secondary")
        .UpperLine(ViewBag.upperline)
        .Period(3)
        .Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Momentum)
        .Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close)
        .SeriesName("Apple Inc").Add();
    })
    .ZoomSettings(zn => zn.EnableSelectionZooming(true)
    .Mode(Syncfusion.EJ2.Charts.ZoomMode.X)
    .Crosshair(cr =>
    cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
    )

```

```

        .PrimaryXAxis(px => px.ZoomFactor(0.2)
        .ZoomPosition(0.6)
        .ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .MajorGridLines(ViewBag.majorGridLines)
        .CrosshairTooltip(ViewBag.crosshairTooltip)
        ).PrimaryYAxis(py => py.RowIndex(1)
        .OpposedPosition(true).Title("Price")
        .Minimum(50).Maximum(170)
        .Interval(30)
        .LabelFormat("{value}M")
        .PlotOffset(25)
        .LineStyle(ViewBag.lineStyle)
        ).Title("AAPL 2012-2017")
        .ChartArea(area => area.Border(ViewBag.ChartBorder))
        .Tooltip(tl => tl.Enable(true).Shared(true))
        .LegendSettings(lg => lg.Visible(false)).Load("load").Render()
</script>
var dataSource = window.chartData;
</script>

```

CUSTOM-MOMENTUM.CS

```

public IActionResult Index()
{
    return View();
}

```

Moving Average Convergence Divergence (MACD)

MACD is based on the difference between two EMA's. To render a MACD Indicator, use indicator [Type](#) as **MACD**. MACD indicator will be represented by MACD line, signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

CSHTML

```

<script src="chart/technical-indicators/macd/financial-data.js"></script>
@Html.EJS().Chart("container-vertical").Rows(rows =>
{
    rows.Height("40%").Add();
    rows.Height("60%").Add();
}).Axes(ax =>
{
    ax.Name("secondary").Title("MACD").OpposedPosition(true).Minimum(-
3.5).Maximum(3.5).MajorTickLines(ViewBag.majorTickLines).MajorGridLines(View
Bag.majorGridLines).Interval(3.5).RowIndex(0).LineStyle(ViewBag.lineStyle).A
dd();
}
).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("
y").High("high").Low("low").Open("open").Close("close").Volume("volume").Bea
rFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name
("Apple Inc").Width(2).Add();
}).Indicators(id =>

```

```

    {
        id.Fill("#6063ff").MacdType(Syncfusion.EJ2.Charts.MacdType.Both).MacdPositiveColor("#2ecd71").MacdNegativeColor("#e74c3d").Fill("#6063ff").YAxisName("secondary").FastPeriod(8).SlowPeriod(5).Period(3).Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Macd).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple Inc").Add();
    }).ZoomSettings(zn =>
zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
).PrimaryXAxis(px =>
px.ZoomFactor(0.2).ZoomPosition(0.6).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines)
).PrimaryYAxis(py =>
py.RowIndex(1).OpposedPosition(true).LabelFormat("${value}M").Title("Price").Minimum(50).Maximum(170).Interval(30).LineStyle(ViewBag.lineStyle).LabelFormat("{value}M").PlotOffset(25)
).Title("AAPL 2012-2017").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Tooltip(tl =>
tl.Enable(true).Shared(true)).LegendSettings(lg =>
lg.Visible(false)).Load("load").Render()
<script>
    var dataSource = window.chartData;
</script>

```

MACD.CS

```

public IActionResult Index()
{
    return View();
}

```

Customization of MACD

stroke, stroke-width, and color of macdLine can be customized by using [MacdLine](#), property of indicator. The positive and negative changes of histogram can be customized by [MacdPositiveColor](#) and [MacdNegativeColor](#) properties. The [MacdType](#) is used to define the type of MACD indicator. To customize the MACD period using [SlowPeriod](#) and [FastPeriod](#) properties.

CSHTML

```

<script src="chart/technical-indicators/macd/financial-data.js"></script>
@Html.EJS().Chart("container-vertical").Rows(rows =>
{
    rows.Height("40%").Add();
    rows.Height("60%").Add();
}).Axes(ax =>
{
    ax.Name("secondary").Title("MACD").OpposedPosition(true).Minimum(-3.5).Maximum(3.5).MajorTickLines(ViewBag.majorTickLines).MajorGridLines(ViewBag.majorGridLines).Interval(3.5).RowIndex(0).LineStyle(ViewBag.lineStyle).Add();
}

```



```

        ).Series(series =>
        {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("y").High("high").Low("low").Open("open").Close("close").Volume("volume").BearFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name("Apple Inc").Width(2).Add();
        }).Indicators(id =>
        {
id.Fill("#6063ff").MacdType(Syncfusion.EJ2.Charts.MacdType.Both).MacdPositiveColor("#2ecd71").MacdNegativeColor("#e74c3d").Fill("#6063ff").YAxisName("secondary").FastPeriod(8).SlowPeriod(5).Period(3).Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Macd).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple Inc").Add();
        }).ZoomSettings(zn =>
zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).CrosShair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
        ).PrimaryXAxis(px =>
px.ZoomFactor(0.2).ZoomPosition(0.6).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines)
        ).PrimaryYAxis(py =>
py.RowIndex(1).OpposedPosition(true).LabelFormat("${value}M").Title("Price").Minimum(50).Maximum(170).Interval(30).LineStyle(ViewBag.lineStyle).LabelFormat("{value}M").PlotOffset(25)
        ).Title("AAPL 2012-2017").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Tooltip(tl =>
tl.Enable(true).Shared(true)).LegendSettings(lg =>
lg.Visible(false)).Load("load").Render()
<script>
    var dataSource = window.chartData;
</script>

```

CUSTOM-MACD.CS

```

public IActionResult Index()
{
    return View();
}

```

Relative Strength Index (RSI)

RSI shows how strongly a stock is moving in its current direction. To render a RSI Indicator, use indicator [Type](#) as [RSI](#). RSI indicator will be represented by three lines (upperBand, lowerBand, signalLine). The upperBand and lowerBand values are customized by [OverBought](#) and [OverSold](#) properties of indicator and the signalLine is calculated by RSI formula.

CSHTML

```

<script src="chart/technical-indicators/rsi/financial-data.js"></script>
@Html.EJS().Chart("container-vertical").Rows(rows =>
{
    rows.Height("40%").Add();
    rows.Height("60%").Add();
}).Axes(ax =>

```

```

{
    ax.Name("secondary").OpposedPosition(true).Title("RSI").Minimum(0).Maximum(120).MajorTickLines(ViewBag.majorTickLines).MajorGridLines(ViewBag.majorGridLines).LineStyle(ViewBag.lineStyle).Interval(60).RowIndex(0).Add();
}
    ).Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("y").High("high").Low("low").Open("open").Close("close").Volume("volume").BearFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name("Apple Inc").Width(2).Add();
    }).Indicators(id =>
    {
        id.Fill("#6063ff").ShowZones(true).OverBought(70).LowerLine(ViewBag.lowerLine).UpperLine(ViewBag.upperLine).OverSold(30).YAxisName("secondary").Period(3).Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Rsi).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple Inc").Add();
    }).ZoomSettings(zn =>
    zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).CrosShair(cr =>
    cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
    ).PrimaryXAxis(px =>
    px.ZoomFactor(0.2).ZoomPosition(0.6).MajorGridLines(ViewBag.majorGridLines).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    ).PrimaryYAxis(py =>
    py.RowIndex(1).OpposedPosition(true).Title("Price").MajorTickLines(ViewBag.majorTickLines).LineStyle(ViewBag.lineStyle).Minimum(50).Maximum(170).Interval(30).LabelFormat("${value}").PlotOffset(25)
    ).Title("AAPL 2012-2017").ChartArea(area =>
    area.Border(ViewBag.ChartBorder)).Tooltip(tl =>
    tl.Enable(true).Shared(true)).LegendSettings(lg =>
    lg.Visible(false)).Load("load").Render()
<script>
    var dataSource = window.chartData;
</script>

```

RSI.CS

```

public IActionResult Index()
{
    return View();
}

```

Simple Moving Average (SMA)

Moving average Indicators are used to define the direction of the trend. To render a SMA Indicator, use indicator [Type](#) as `Sma`.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
```

```

{
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("y").High("high").Low("low").Open("open").Close("close").Volume("volume").BearFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name("Apple Inc").Width(2).Add();
}).Indicators(id =>
{
id.Fill("#6063ff").Period(14).Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Sma).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple Inc").Add();
}).ZoomSettings(zn =>
zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
).PrimaryXAxis(px =>
px.ZoomFactor(0.2).ZoomPosition(0.6).CrosshairTooltip(ViewBag.crosshairTooltip).MajorGridLines(ViewBag.majorGridLines).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
).PrimaryYAxis(py =>
py.Title("Price").Minimum(50).Maximum(170).Interval(30).LineStyle(ViewBag.lineStyle).LabelFormat("{value}M")
).Title("AAPL 2012-2017").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Tooltip(tl =>
tl.Enable(true).Shared(true)).LegendSettings(lg =>
lg.Visible(false)).Load("load").Render()
<script>
    var dataSource = window.chartData;
</script>

```

SMA.CS

```

public IActionResult Index()
{
    return View();
}

```

Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator [Type](#) as [Stochastic](#). stochastic indicator will be represented by four lines (upperLine, lowerLine, periodLine, signalLine). In stochastic indicator the upperBand value and lowerBand value is customized by [OverBought](#) and [OverBought](#) properties of indicators and the periodLine and signalLine is render based on stochastic formula.

CSHTML

```

@Html.EJS().Chart("container-vertical").Rows(rows =>
{
    rows.Height("40%").Add();
    rows.Height("60%").Add();
}).Axes(ax =>
{
ax.Name("secondary").OpposedPosition(true).Minimum(0).Maximum(120).LineStyle

```

```

(ViewBag.lineStyle).MajorGridLines(ViewBag.majorGridLines).MajorTickLines(Vi
ewBag.majorTickLines)
.Interval(60).RowIndex(0).Add();
    }
    ).Series(series =>
    {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("
y").High("high").Low("low").Open("open").Close("close").Volume("volume").Bea
rFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name
("Apple Inc").Width(2).Add();
    }).Indicators(id =>
    {
id.Fill("#6063ff").KPeriod(2).DPeriod(3).YAxisName("secondary").Period(3).Lo
werLine(ViewBag.lowerLine).UpperLine(ViewBag.upperLine).PeriodLine(ViewBag.p
eriodLine).Animation(ViewBag.animation).Type(Syncfusion.EJ2.Charts.Technical
Indicators.Stochastic).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close
).SeriesName("Apple Inc").Add();
    }).ZoomSettings(zn =>
zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X).Cros
shair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
    ).PrimaryXAxis(px =>
px.ZoomFactor(0.2).ZoomPosition(0.6).ValueType(Syncfusion.EJ2.Charts.ValueTy
pe.DateTime).MajorGridLines(ViewBag.majorGridLines)
    ).ChartArea(area =>
area.Border(ViewBag.ChartBorder)).PrimaryYAxis(py =>
py.RowIndex(1).OpposedPosition(true).Title("Price").LineStyle(ViewBag.lineSt
yle).Minimum(80).Maximum(170).Interval(30).LabelFormat("{value}M").PlotOffse
t(25)
    ).Title("AAPL 2012-2017").Tooltip(tl =>
tl.Enable(true).Shared(true)).LegendSettings(lg =>
lg.Visible(false)).Load("load").Render()
<script>
    var dataSource = window.chartData;
</script>

```

STOCHASTIC.CS

```

public IActionResult Index()
{
    return View();
}

```

Customization of StochasticIndicator

stroke, stroke-width, and color of upperLine can be customized by using, [UpperLine](#), the lowerLine can be customized by using [LowerLine](#) and the periodLine can be customized by using [PeriodLine](#) properties of indicator. To customize the period to find the average price using [KPeriod](#) and [DPeriod](#) properties.

CSHTML

```

<script src="chart/technical-indicators/stochastic/financial-
data.js"></script>
@Html.EJS().Chart("container-vertical").Rows(rows =>

```

```

        {
            rows.Height("40%").Add();
            rows.Height("60%").Add();
        }).Axes(ax =>
        {
            ax.Name("secondary").OpposedPosition(true).Minimum(0).Maximum(120).LineStyle(
                ViewBag.lineStyle).MajorGridLines(ViewBag.majorGridLines).MajorTickLines(
                ViewBag.majorTickLines).Interval(60).RowIndex(0).Add();
        })
        .Series(series =>
        {
            series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("y")
                .High("high").Low("low").Open("open").Close("close").Volume("volume")
                .BearFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource")
                .Name("Apple Inc").Width(2).Add();
        }).Indicators(id =>
        {
            id.Fill("#6063ff").KPeriod(2).DPeriod(3).YAxisName("secondary").Period(3)
                .LowerLine(ViewBag.lowerLine).UpperLine(ViewBag.upperLine).PeriodLine(
                ViewBag.periodLine).Animation(ViewBag.animation).Type(Syncfusion.EJ2.Charts.Technical
                Indicators.Stochastic).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close)
                .SeriesName("Apple Inc").Add();
        }).ZoomSettings(zn =>
            zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X).Cros
            shair(cr =>
                cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
                ).PrimaryXAxis(px =>
                    px.ZoomFactor(0.2).ZoomPosition(0.6).ValueType(Syncfusion.EJ2.Charts.ValueTy
                    pe.DateTime).MajorGridLines(ViewBag.majorGridLines)
                    ).ChartArea(area =>
                        area.Border(ViewBag.ChartBorder).PrimaryYAxis(py =>
                            py.RowIndex(1).OpposedPosition(true).Title("Price").LineStyle(
                                ViewBag.lineStyle).Minimum(80).Maximum(170).Interval(30)
                                .LabelFormat("{value}M").PlotOffset(25)
                                ).Title("AAPL 2012-2017").Tooltip(tl =>
                                    tl.Enable(true).Shared(true)).LegendSettings(lg =>
                                        lg.Visible(false)).Load("load").Render()
                                })
                    )
                )
            )
        }
    }
</script>
    var dataSource = window.chartData;
</script>

```

CUSTOM-STOCHASTIC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Charts;
namespace EJ2MVCSampleBrowser.Controllers.Chart
{

```

```

public partial class ChartController : Controller
{
    public ActionResult Stochastic()
    {
        ViewBag.crosshairTooltip = new { enable = true };
        ViewBag.majorGridLines = new { width = 0.00001 };
        ViewBag.lineStyle = new { width = 0.00001 };
        ViewBag.animation = new { enable = false };
        ViewBag.majorTickLines = new { width = 0.00001 };
        ViewBag.upperLine = new { color = "#0f0000" };
        ViewBag.lowerLine = new { color = "#080000" };
        ViewBag.periodLine = new { color = "#f2ec2f" };
        ViewBag.chartBorder = new ChartBorder { Color = "transparent" };
        return View();
    }
}

```

Triangular Moving Average (TMA)

Moving average indicators are used to define the direction of the trend. To render a TMA Indicator, use indicator [Type](#) as **TMA**.

CSHTML

```

<script src="chart/technical-indicators/tma/financial-data.js"></script>
@Html.EJS().Chart("container-vertical").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).XName("x").YName("y").Animation(ViewBag.animation)
    .High("high").Low("low").Open("open").Close("close").Volume("volume").BearFillColor("#2ecd71").BullFillColor("#e74c3d").DataSource("dataSource").Name("Apple Inc").Width(2).Add();
    }).Indicators(id =>
    {
        id.Fill("#6063ff").Period(3).Type(Syncfusion.EJ2.Charts.TechnicalIndicators.Tma).Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple Inc").Add();
    }).ZoomSettings(zn =>
    zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X).Crosshair(cr =>
    cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
    ).PrimaryXAxis(px =>
    px.ZoomFactor(0.2).ZoomPosition(0.6).CrosshairTooltip(ViewBag.crosshairTooltip).MajorGridLines(ViewBag.majorGridLines)
    ).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    ).PrimaryYAxis(py =>
    py.Title("Price").Minimum(50).Maximum(170).Interval(30)
    ).LineStyle(ViewBag.lineStyle).LabelFormat("{value}M")
    ).Title("AAPL 2012-2017").ChartArea(area =>
    area.Border(ViewBag.ChartBorder).Tooltip(tl =>
    tl.Enable(true).Shared(true)).LegendSettings(lg =>
    lg.Visible(false)).Load("load").Render()
    )
    <script>
        var dataSource = window.chartData;

```

```
</script>
```

TMA.CS

```
public IActionResult Index()
{
    return View();
}
```

Customization of Technical Indicators

stroke, stroke-width, and color of signalLine can be customized by using [Fill](#), [Width](#) and [DashArray](#) properties and the [Period](#) property is used to predict the data forecast calculations. The [Field](#) value is used to compare the current price with previous price. It is applicable to Bollinger bands and moving averages. The [ShowZones](#) property is used to show/hides the overBought and overSold regions. It is applicable for RSI and stochastic indicators.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()
```

CUSTOM-TMA.CS

```
public IActionResult Index()
{
    return View();
}
```

Data Source

Usually technical indicators are added along with a financial series. The [SeriesName](#) represents the series, the data of which has to be analysed through indicators.

Technical indicators can also be added without series using [DataSource](#) property of indicator.

CSHTML

```
<script src="chart/technical-indicators/ad/financial-data.js"></script>
@Html.EJS().Chart("container-vertical").Rows(rows =>
{
    rows.Height("40%").Add();
    rows.Height("60%").Add();
}).Axes(ax =>
```

```

{
    ax.Name("secondary").
    OpposedPosition(true).
    Minimum(-7000000000).
    Maximum(5000000000).
    Interval(6000000000).
    Title("Accumulation
Distribution").MajorTickLines(ViewBag.majorTickLines).LineStyle(ViewBag.line
Style).
    RowIndex(0).Add();
}

).Indicators(id =>
{
id.Fill("#6063ff").KPeriod(2).DPeriod(3).YAxisName("secondary").Period(3).Ty
pe(Syncfusion.EJ2.Charts.TechnicalIndicators.AccumulationDistribution).Field
(Syncfusion.EJ2.Charts.FinancialDataFields.Close).SeriesName("Apple
Inc").Add();
}).ZoomSettings(zn =>
zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X).Cros
shair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true)
).PrimaryXAxis(px => {

px.ZoomFactor(0.2).MajorGridLines(dataT).ZoomPosition(0.6).MajorGridLines(Vi
ewBag.majorGridLines).CrosshairTooltip(ViewBag.crosshairTooltip).ValueType(S
yncfusion.EJ2.Charts.ValueType.DateTime);
}).PrimaryYAxis(py =>
py.RowIndex(1).OpposedPosition(true).LineStyle(ViewBag.lineStyle).Title("Pri
ce").Minimum(80).Maximum(170).Interval(30).LabelFormat("${value}").PlotOffse
t(25)
<script>
    var dataSource = window.chartData;
</script>

```

DATASOURCE.CS

```

public IActionResult Index()
{
    return View();
}

```

<!-- markdownlint-disable MD036 -->

Trend Lines in ASP.NET MVC Chart Component

Trendlines are used to show the direction and speed of price.

Trendlines can be generated for Cartesian type series (Line, Column, Scatter, Area, Candle, Hilo etc.) except bar type series. You can add more than one trendline to a series.

Chart supports 6 types of trendlines.

Linear

A linear trendline is a best fit straight line that is used with simpler data sets. To render a linear trendline, use trendline [Type](#) as `Linear`.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
    .Name("Rupees")
    .Width(2).Add(); })
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines))
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

LINEAR.CS

```

public IActionResult Index()
{
    return View();
}

```

Exponential

An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render a exponential trendline, use trendline [Type](#) as **Exponential**.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
        .Name("Rupees")
        .Width(2).Add();
    }).PrimaryXAxis(px =>
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines)
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>
```

EXPONENTIAL.CS

```
public IActionResult Index()
{
    return View();
}
```

Logarithmic

A logarithmic trendline is a best-fit curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out.

A logarithmic trendline can use negative and/or positive values.

To render a logarithmic trendline, use trendline [Type](#) as **Logarithmic**.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
    .Name("Rupees")
    .Width(2).Add(); })
    .PrimaryXAxis(px =>
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines))
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>
```

LOGARITHMIC.CS

```
public IActionResult Index()
{
    return View();
}
```

Polynomial

A polynomial trendline is a curved line that is used when data fluctuates.

To render a polynomial trendline, use trendline [Type](#) as **Polynomial**.

PolynomialOrder used to define the polynomial value.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
    .Name("Rupees")
    .Width(2).Add(); })
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines)
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
```

```
];
</script>
```

POLYNOMIAL.CS

```
public IActionResult Index()
{
    return View();
}
```

Power

A power trendline is a curved line that is best used with data sets that compare measurements that increase at a specific rate.

To render a power trendline, use trendline [Type](#) as **Power**.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
        .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter)
        .XName("x").YName("y")
        .Marker(ViewBag.marker)
        .DataSource("powerData")
        .Name("Rupees")
        .Width(2).Add(); })
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
.MajorGridLines(ViewBag.majorGridLines)
.PrimaryYAxis(py => py.Title("Rupees against Dollars")
.Interval(10)
.MajorTickLines(ViewBag.majorTickLines)
.LineStyle(ViewBag.lineStyle))
.ChartArea(area => area.Border(ViewBag.ChartBorder))
.Title("Historical Indian Rupee Rate (INR USD)")
.LegendSettings(lg => lg.Visible(false))
.Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
```

```

        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

POWER.CS

```

public IActionResult Index()
{
    List<powerData> chartData = new List<powerData>
    {
        new powerData { x= 1, y= 10 },
        new powerData { x= 2, y= 50 },
        new powerData { x= 3, y= 80 },
        new powerData { x= 4, y= 110 },
        new powerData { x= 5, y= 180 },
        new powerData { x= 6, y= 220 },
        new powerData { x= 7, y= 300 },
        new powerData { x= 8, y= 370 },
        new powerData { x= 9, y= 490 },
        new powerData { x= 10, y= 500 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class powerData
{
    public double x;
    public double y;
}

```

Moving Average

A moving average trendline smoothen out fluctuations in data to show a pattern or trend more clearly.

To render a moving average trendline, use trendline [Type](#) as `MovingAverage`.

`Period` property defines the period to find the moving average.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("powerData")
    .Name("Rupees")
    .Width(2).Add(); })
    .PrimaryXAxis(px =>
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines))
    .PrimaryYAxis(py => py.Title("Rupees against Dollars"))

```

```

        .Interval(10)
        .MajorTickLines(ViewBag.majorTickLines)
        .LineStyle(ViewBag.lineStyle))
        .ChartArea(area => area.Border(ViewBag.ChartBorder))
        .Title("Historical Indian Rupee Rate (INR USD)")
        .LegendSettings(lg => lg.Visible(false))
        .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

MOVINGAVERAGE.CS

```

public IActionResult Index()
{
    return View();
}

```

Customization of Trendlines

The [Fill](#) and [Width](#) properties are used to customize the appearance of the trendline.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("powerData")
    .Name("Rupees")
    .Width(2).Add(); })
    .PrimaryXAxis(px =>
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)

```

```

        .MajorGridLines(ViewBag.majorGridLines))
        .PrimaryYAxis(py => py.Title("Rupees against Dollars")
        .Interval(10)
        .MajorTickLines(ViewBag.majorTickLines)
        .LineStyle(ViewBag.lineStyle))
        .ChartArea(area => area.Border(ViewBag.ChartBorder))
        .Title("Historical Indian Rupee Rate (INR USD)")
        .LegendSettings(lg => lg.Visible(false))
        .Tooltip(tt => tt.Enable(true)).Render()

<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

CUSTOM-TRENDLINE.CS

```
public IActionResult Index()
{
    return View();
}
```

Forecasting

Trendlines forecasting is the prediction of future/past situations.

Forecasting can be classified into two types as follows

- Forward Forecasting
- Backward Forecasting

Forward Forecasting

The value set for `forwardForecast` is used to determine the distance moving towards the future trend.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
    .Name("Rupees")
    .Width(2).Add(); })
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines)
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

FORWARD-FORECAST.CS

```

public IActionResult Index()
{
    return View();
}

```

Backward Forecasting

The value set for the backwardForecast is used to determine the past trends.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
    .Name("Rupees")
    .Width(2).Add(); })
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines)
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

BACKWARD-FORECAST.CS

```

public IActionResult Index()
{
    return View();
}

```

Show or hide a trendline

You can show or hide the trendline by setting trendline `visible` property.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Trendlines(ViewBag.trendLines)
    .Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .XName("x").YName("y")
    .Marker(ViewBag.marker)
    .DataSource("series1")
    .Name("Rupees")
    .Width(2).Add(); })
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    .MajorGridLines(ViewBag.majorGridLines)
    .PrimaryYAxis(py => py.Title("Rupees against Dollars")
    .Interval(10)
    .MajorTickLines(ViewBag.majorTickLines)
    .LineStyle(ViewBag.lineStyle))
    .ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Title("Historical Indian Rupee Rate (INR USD)")
    .LegendSettings(lg => lg.Visible(false))
    .Trendlines(tl => tl.Visible(false))
    .Tooltip(tt => tt.Enable(true)).Render()
<script type="text/javascript">
    var series1 = [];
    var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68,
9.48, 10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52,
36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17,
41.20, 43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
    var point1;
    var i;
    var j = 0;
    for (i = 1973; i <= 2013; i++) {
        point1 = { x: i, y: yValue[j] };
        series1.push(point1);
        j++;
    }
    var powerData = [
        { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y:
110 },
        { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y:
370 }, { x: 9, y: 490 }, { x: 10, y: 500 }
    ];
</script>

```

HIDE.CS

```

public IActionResult Index()
{
    return View();
}

```

Data Markers in ASP.NET MVC Chart Component

Data markers are used to provide information about the data points in the series. You can add a shape to adorn each data point.

<!-- markdownlint-disable MD036 -->

Marker

<!-- markdownlint-disable MD036 -->

Markers can be added to points by enabling the [Visible](#) option of the marker property. By default, distinct markers will be enabled for each series in the chart.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
    Marker(mr=>mr.Visible(true))
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
    Marker(mr=>mr.Visible(true))
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    Width(2).Add();
}).
Title("Olympic Medal Counts - RIO").Render()
```

MARKER.CS

```
public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {
        new VerticalData{ x= 2005, y= 28, y1= 18 },
        new VerticalData{ x= 2006, y= 25, y1= 10 },
        new VerticalData{ x= 2007, y= 26, y1= 20 },
        new VerticalData{ x= 2008, y= 27, y1= 35 },
        new VerticalData{ x= 2009, y= 32, y1= 23 },
        new VerticalData{ x= 2010, y= 35, y1= 25 },
        new VerticalData{ x= 2011, y= 30, y1= 15 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class VerticalData
{
    public double x;
    public double y;
    public double y1;
}
```

Shape

Markers can be assigned with different shapes such as Rectangle, Circle, Diamond etc using the [Shape](#) property.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()

```

SHAPE.CS

```

public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {
        new VerticalData{ x= 2005, y= 28 },
        new VerticalData{ x= 2006, y= 25 },
        new VerticalData{ x= 2007, y= 26 },
        new VerticalData{ x= 2008, y= 27 },
        new VerticalData{ x= 2009, y= 32 },
        new VerticalData{ x= 2010, y= 35 },
        new VerticalData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class VerticalData
{
    public double x;
    public double y;
}

```

Note: To know more about the marker shape type refer the [Shape](#).

Images

Apart from the shapes, you can also add custom images to mark the data point using the [ImageUrl](#) property.

CSHTML

```

@ControlSection {
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).

```

```

    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").Render()
}

```

IMAGES.CS

```

public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {
        new VerticalData{ x= 2005, y= 28 },
        new VerticalData{ x= 2006, y= 25 },
        new VerticalData{ x= 2007, y= 26 },
        new VerticalData{ x= 2008, y= 27 },
        new VerticalData{ x= 2009, y= 32 },
        new VerticalData{ x= 2010, y= 35 },
        new VerticalData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class VerticalData
{
    public double x;
    public double y;
}

```

Customization

Marker's color and border can be customized using **Fill** and **Border** properties.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").Render()

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {
        new VerticalData{ x= 2005, y= 28 },

```

```

        new VerticalData{ x= 2006, y= 25 },
        new VerticalData{ x= 2007, y= 26 },
        new VerticalData{ x= 2008, y= 27 },
        new VerticalData{ x= 2009, y= 32 },
        new VerticalData{ x= 2010, y= 35 },
        new VerticalData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class VerticalData
{
    public double x;
    public double y;
}

```

Customizing specific point

You can also customize the specific marker and label using [PointRender](#) event. The `PointRender` event allows you to change the shape, color and border for a point.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").PointRender("point").Render()
<script>
function point(args) {
    if(args.point.index === 1) {
        args.shape = "Circle"
    }
}
</script>

```

CUSTOM-POINT.CS

```

public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {
        new VerticalData{ x= 2005, y= 28 },
        new VerticalData{ x= 2006, y= 25 },
        new VerticalData{ x= 2007, y= 26 },
        new VerticalData{ x= 2008, y= 27 },
        new VerticalData{ x= 2009, y= 32 },
    }
}

```

```

        new VerticalData{ x= 2010, y= 35 },
        new VerticalData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class VerticalData
{
    public double x;
    public double y;
}

```

Fill marker with series color

Marker can be filled with the series color by setting the [IsFilled](#) property to **true**.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
    Marker(mr=>mr.Visible(true).IsFilled(true)).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").PointRender("point").Render()

```

ISFILLED.CS

```

public ActionResult Index()
{
    List<VerticalData> chartData = new List<VerticalData>
    {
        new VerticalData{ x= 2005, y= 28 },
        new VerticalData{ x= 2006, y= 25 },
        new VerticalData{ x= 2007, y= 26 },
        new VerticalData{ x= 2008, y= 27 },
        new VerticalData{ x= 2009, y= 32 },
        new VerticalData{ x= 2010, y= 35 },
        new VerticalData{ x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class VerticalData
{
    public double x;
    public double y;
}

```


Data Labels in ASP.NET MVC Chart

Data label can be added to a chart series by enabling the [Visible](#) option in the dataLabel. By default, the labels will arrange smartly without overlapping.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(mr=> mr.Visible(true).DataLabel(dl=>dl.Visible(true))).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource). Name("Gold").Width(2).Add();
}).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("O
lympic Medal Counts - RIO").Render()
```

DATALABEL.CS

```
public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}
```

Position

Using [Position](#) property, you can place the label either on **Top**, **Middle**, **Bottom** or **Outer** (outer is applicable for column and bar type series).

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true).Position(Syncfusion.EJ2.Cha
rts.LabelPosition.Middle))).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
```

```

        Name("Gold").
        Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("O
lympic Medal Counts - RIO").Render()

```

POSITION.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}

```

Note: The position `Outer` is applicable for column and bar type series.

Data Label Template

Label content can be formatted by using the template option. Inside the template, you can add the placeholder text `${point.x}` and `${point.y}` to display corresponding data points x & y value. Using [Template](#) property, you can set data label template in chart.

CSHTML

```

<div id="ControlRegion">
    @Html.EJS().Chart("container").Series(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Middle).Template("#template"))).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("O
lympic Medal Counts - RIO").Render()
<script id="template">

```

```
<div style="background:#f5f5f5; border: 1px solid black; padding: 3px 3px 3px 3px">
    <div>${point.x}</div>
    <div>${point.y}</div>
</div>
</script>
```

TEMPLATE.CS

```
public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}
```

Text Mapping

Text from the data source can be mapped using **Name** property.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    @Html.EJS().Chart("container").Series(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true).Name("text"))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("O
lympic Medal Counts - RIO").Render()
```

MAPPING.CS

```
public ActionResult Index()
{
```

```

        List<Data> chartData = new List<Data>
        {
            new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
            new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
            new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
            new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class Data
    {
        public string x;
        public double y;
        public string fill;
        public string text;
    }

```

Format

Data label for the chart can be formatted using [Format](#) property. You can use the global formatting options, such as 'n', 'p', and 'c'.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    @Html.EJS().Chart("container").Series(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true).Format("n2"))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("O
lympic Medal Counts - RIO").Render()

```

FORMAT.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

```

```
public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}
```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Margin

Margin for data label can be applied to using **Left**, **Right**, **Bottom** and **Top** properties.

CSHTML

```
@{
    var border = new { width = 2, color = "red" };
    var margin = new { left = 15, right = 15, bottom = 15, top = 15 };
}
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true).Border(border).Margin(margin)))
        .
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("Olympic Medal Counts - RIO").Render()
```

MARGIN.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}

```

Customization

stroke and border of data label can be customized using Fill and Border properties. Rounded corners can be customized using Rx and Ry properties.

CSHTML

```

@{
    var border = new { width = 2, color = "red" };
    var margin = new { left = 15, right = 15, bottom = 15, top = 15 };
}

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true).Border(border).Name("text")
.Rx(10).Ry(10).Fill("skyblue"))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("O
lympic Medal Counts - RIO").Render()

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
    }
}

```

```

        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}

```

Note: Rx and Ry properties requires Border values not to be null.

Customizing Specific Point

You can also customize the specific marker and label using [PointRender](#) and [TextRender](#) event.

PointRender event allows you to change the shape, color and border for a point, whereas the **TextRender** event allows you to change the text for the point.

CSHTML

```

@Html.EJS().Chart("container").TextRender("textRender").PointRender("pointRender").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr=>
mr.Visible(false).DataLabel(dl=>dl.Visible(true))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Name("Gold").
        Width(2).Add();
    }).PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Title("Olympic Medal Counts - RIO").Render()
<script>
pointRender = function (args) {
    if (args.point.index === 2) {
        args.fill = 'red'
    }
},
textRender = function (args) {
    if (args.point.index === 2) {
        args.text = 'Maximum Temperature';
        args.color = 'red';
    }
    else {
        args.cancel = true;
    }
}
</script>

```

CUSTOM-POINT.CS

```

public ActionResult Index()
{
    List<Data> chartData = new List<Data>
    {
        new Data{ x= "Jan", y= 3, fill= "#498fff", text= "January" },
        new Data{ x= "Feb", y= 3.5, fill= "#ffa060", text= "February" },
        new Data{ x= "Mar", y= 7, fill= "#ff68b6", text= "March" },
        new Data{ x= "Apr", y= 13.5, fill= "#81e2a1", text= "April" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class Data
{
    public string x;
    public double y;
    public string fill;
    public string text;
}

```

Show percentage based on each series points

You can calculate the percentage value based on the sum for each series using the **SeriesRender** and **TextRender** events in the chart. In **SeriesRender** calculate the sum of each series y values and In **TextRender** calculate percentage value based on the sum value and modify the text.

CSHTML

```

<div class="control-section">
    <div style="text-align:center">

@Html.EJS().Chart("container").TextRender("onTextRender").SeriesRender("onSeriesRender").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .Marker(mr => mr.DataLabel(dl => dl.Visible(true).Font(ff =>
ff.FontWeight("600").Color("#ffffff").Position(Syncfusion.EJ2.Charts.LabelPosition.Top))).XName("x")

.YName("yValue").DataSource(ViewBag.dataSource1).Name("Gold").Width(2).Add();
;
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .Marker(mr => mr.DataLabel(dl => dl.Visible(true).Font(ff =>
ff.FontWeight("600").Color("#ffffff").Position(Syncfusion.EJ2.Charts.LabelPosition.Top))).XName("x").YName("yValue1").DataSource(ViewBag.dataSource2)
    .Name("Silver").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .Marker(mr => mr.DataLabel(dl => dl.Visible(true).Font(ff =>
ff.FontWeight("600").Color("#ffffff").Position(Syncfusion.EJ2.Charts.LabelPosition.Top))).XName("x")

.YName("yValue2").DataSource(ViewBag.dataSource3).Name("Brozen").Width(2).Add();
    }) .PrimaryYAxis(px => px.LabelStyle(ls =>
ls.Color("transparent").LineStyle(ls => ls.Width(0)).MajorTickLines(mg =>
mg.Width(0))

```



```

        .MajorGridLines(mg => mg.Width(0))
        ).PrimaryXAxis(px => px.Interval(1))
        .ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg =>
mg.Width(0))
        ).Tooltip(tt => tt.Enable(true)).ChartArea(area => area.Border(br =>
br.Color("transparent")))
        ).LegendSettings(lg => lg.Visible(false)).Title("Olympic Medal Counts -
RIO").Render()
    </div>
</div>
<script>
    var total = [];
    function onSeriesRender(args) {
        for (var i = 0; i < args.data.length; i++) {
            if (!total[args.data[i].x]) total[args.data[i].x] = 0;
            total[args.data[i].x] = total[args.data[i].x] +
parseInt(args.data[i].y);
        }
    }
    function onTextRender(args) {
        var percentage = (parseInt(args.text) / total[args.point.x]) * 100;
        percentage = percentage % 1 === 0 ? percentage :
percentage.toFixed(2);
        args.text = percentage + "%";
    }
</script>

```

SERIES-PERCENTAGE.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData1 = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", y = 46 },
        new ColumnChartData { x= "GBR", y = 27 },
        new ColumnChartData { x= "CHN", y = 26 }
    };
    ViewBag.dataSource1 = chartData1;
    List<ColumnChartData> chartData2 = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", y = 37 },
        new ColumnChartData { x= "GBR", y = 23 },
        new ColumnChartData { x= "CHN", y = 18 }
    };
    ViewBag.dataSource2 = chartData2;
    List<ColumnChartData> chartData3 = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", y = 38 },
        new ColumnChartData { x= "GBR", y = 17 },
        new ColumnChartData { x= "CHN", y = 26 }
    };
    ViewBag.dataSource3 = chartData3;
    return View();
}

public class ColumnChartData
{

```

```

        public string x;
        public double y;
    }

```

Annotation

Annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

<!-- markdownlint-disable MD033 -->

You can add annotations to the chart by using the `<code>annotations</code> option. By using the Content option of annotation object, you can specify the id of the element that needs to be displayed in the chart area.`

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
Annotations(an => {

an.X("USA").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti
calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content).Ad
d();

an.X("GBR").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti
calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content1).A
dd();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()

```

ANNOTATION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
    }
}

```

```

        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Region

Annotations can be placed either with respect to **Series** or **Chart**. by default, it will placed with respect to **Chart**.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    Annotations(an => {

an.X("USA").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti
calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content).Ad
d();

an.X("GBR").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti
calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content1).A
dd();
    }).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").Render()

```

REGION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {

```

```

        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Co-ordinate Units

Specified the coordinates units of the annotation either **Pixel** or **Point**.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    Annotations(an => {

an.X("USA").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti
calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content).Ad
d();

an.X("GBR").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti
calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content1).A
dd();
    }).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").Render()

```

CO-ORDINATE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Alignment

Annotation provides **VerticalAlignment** and **HorizontalAlignment**. The **VerticalAlignment** can be customized via **Top**, **Bottom** or **Middle** and the **horizontalAlignment** can be customized via **Near**, **Far** or **Center**.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
Marker(ViewBag.Marker).
XName("x").
YName("yValue").
DataSource(ViewBag.dataSource).
Name("Gold").
Width(2).Add();
}).
Annotations(an => {
an.X("USA").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Verti

```

```

calAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content).Add();

an.X("GBR").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).VerticalAlignment(Syncfusion.EJ2.Charts.Position.Top).Content(ViewBag.content1).Add();
    }).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").Render()

```

ALIGNMENT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Adding y-axis sub title through on annotation

By setting text div in the **Content** option of annotation object you can add sub title to chart y-axis. Specified the **Coordinate** value as **Pixel** and customize x and y location of the text.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area => area.Border(br =>
br.Color("transparent"))).Series(series =>
{

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue")
    .Marker(mr => mr.Visible(true).Width(10).Height(10)).YName("yValue")
    .DataSource(ViewBag.dataSource).Name("Germany").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
).PrimaryYAxis(py => py.Title("(m3/min)").Title("Inflation - Consumer
Price").Tooltip(tt => tt.Enable(true)).Annotations(an =>
{
an.Region(Syncfusion.EJ2.Charts.Regions.Chart).X("2").Y("220").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Pixel).
Content("#templateWrap").Add();
}).Render()
<script id="templateWrap" type="text/x-template">
    <div id="text" style="transform: rotate(-90deg);">Speed Rate</div>
</script>

```

AXIS-SUB.CS

```

public ActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = "2014", yValue = 21 },
        new LineChartData { xValue = "2015", yValue = 24 },
        new LineChartData { xValue = "2016", yValue = 36 },
        new LineChartData { xValue = "2017", yValue = 38 },
        new LineChartData { xValue = "2018", yValue = 54 },
        new LineChartData { xValue = "2019", yValue = 57 },
        new LineChartData { xValue = "2020", yValue = 70 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public string xValue;
    public double yValue;
    public double yValue1;
}

```

Legend and in ASP.NET MVC Chart Component

Legend provides information about the series rendered in the chart.

Enable Legend

You can use legend for the chart by setting the [Visible](#) property to true in [LegendSettings](#) object.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
    )
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render();

```

LEGEND.CS

```

public IActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
        new LineChartData { xValue = new DateTime(2009, 01, 01),
yValue = 54, yValue1 = 66 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 57, yValue1 = 78 },
        new LineChartData { xValue = new DateTime(2011, 01, 01),
yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

Position and Alignment

By using the [Position](#) property, you can position the legend at left, right, top or bottom of the chart. The legend is positioned at the bottom of the chart, by default.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).Title("Inflation - Consumer Price").Load("load").Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}
```

Custom position helps you to position the legend anywhere in the chart using x, y coordinates.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend =>
legend.Position(Syncfusion.EJ2.Charts.LegendPosition.Top)
).Title("Inflation - Consumer Price").Load("load").Render()
```

POSITION.CS

```
public IActionResult Index()
{
    List chartData = new List
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
        new LineChartData { xValue = new DateTime(2009, 01, 01),
yValue = 54, yValue1 = 66 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 57, yValue1 = 78 },
        new LineChartData { xValue = new DateTime(2011, 01, 01),
yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}
```

Legend Reverse

You can reverse the order of the legend items by using the [Reverse](#) property. By default, legend for the first series in the collection will be placed first.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x").YName("yValue").DataSource(ViewBag.dataSource)
    .Name("Gold").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x").YName("yValue1").DataSource(ViewBag.dataSource)
    .Name("Silver").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .XName("x").YName("yValue2").DataSource(ViewBag.dataSource)
    .Name("Brozen").Width(2).Add();
}).PrimaryYAxis(px =>
px.LabelStyle(ls=>ls.Color("transparent")).LineStyle(ls=>ls.Width(0)).MajorT
ickLines(mg=>mg.Width(0))
    .MajorGridLines(mg=>mg.Width(0))
    ).PrimaryXAxis(px => px.Interval(1)

.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg=>mg.W
idth(0))
    ).ChartArea(area => area.Border(br=>br.Color("transparent")))
    ).LegendSettings(lg => lg.reverse(true)).Title("Olympic Medal Counts
- RIO").Load("load").Render()
```

REVERSE.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46, yValue1=37,
yValue2=38 },
        new ColumnChartData { x= "GBR", yValue= 27, yValue1=23,
yValue2=17 },
        new ColumnChartData { x= "CHN", yValue= 26, yValue1=18,
yValue2=26 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
    public double yValue1;
    public double yValue2;
}
```

<!-- markdownlint-disable MD036 -->

Legend Alignment

<!-- markdownlint-disable MD036 -->

You can align the legend as **Center**, **Far** or **Near** to the chart using [Alignment](#) property.

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
    ).LegendSettings(legend =>
legend.Alignment(Syncfusion.EJ2.Charts.Alignment.Far)
    ).Title("Inflation - Consumer Price").Load("load").Render()
```

ALIGNMENT.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
```

```

        public double bronze;
    }

```

Customization

To change the legend icon shape, you can use [LegendShape](#) property in the [Series](#). By default legend icon shape is [SeriesType](#).

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend =>
legend.shape(Syncfusion.EJ2.Charts.LegendShape.Square)
).Title("Inflation - Consumer Price").Load("load").Render()

```

CUSTOM.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

```

```
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}
```

Legend Size

By default, legend takes 20% - 25% of the chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the chart's width vertically, when placed on left or right position of the chart. You can change this default legend size by using the [Width](#) and [Height](#) property of the [LegendSettings](#).

CSHTML

```
@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend => legend.Size("45%")
).Title("Inflation - Consumer Price").Load("load").Render()
```

SIZE.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
    }
```

```

        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend Item Size

You can customize the size of the legend items by using the [ShapeHeight](#) and [ShapeWidth](#) property.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend => legend.ShapeWidth("10")
).Title("Inflation - Consumer Price").Load("load").Render()

```

ITEM-SIZE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
    }
}

```

```

        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Paging for Legend

Paging for Legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. You can view each legend items by navigating between the pages using navigation buttons.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}) .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
) .PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
) .LegendSettings(legend => legend.Width(50)
) .Title("Inflation - Consumer Price").Load("load").Render()

```

PAGING.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
    }
}

```



```

        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend Text Wrap

When the legend text exceed the container, the text can be wrapped by using the [textWrap](#) property. End user can also wrap the legend text based on the [maximumLabelWidth](#) property.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("country").YName("gold").DataSource(ViewBag.dataSource).Name("Gold Medals").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("country").YName("silver").DataSource(ViewBag.dataSource).Name("Silver Medals").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("country").YName("bronze").DataSource(ViewBag.dataSource).Name("Bronze Medals").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).LegendSettings(legend =>
legend.shape(Syncfusion.EJ2.Charts.LegendShape.Circle).position(@Syncfusion.EJ2.Charts.LegendPosition.Right)

.maximumLabelWidth(50).textWrap(Syncfusion.EJ2.Charts.TextWrap.Wrap)).Title("Olympic Medals").Render();

```

TEXTWRAP.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Set the label color based on series color

You can set the legend label color based on series color by using chart's [Loaded](#) event.

CSHTML

```

@Html.EJS().Charts("charts").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).Title("Olympic Medals").Loaded("onChartLoaded").Render()

<script type="text/javascript">

```

```

var colors = ['#00BDAE', '#404041', '#357CD2'];
function onChartLoaded(args) {
    let chart = document.querySelector('.e-chart');
    let legendTextCol =
chart.querySelectorAll('[id*="chart_legend_text_"]');
    for (let i = 0; i < legendTextCol.length; i++) {
        //set the color to legend label
        legendTextCol[i].setAttribute('fill', colors[i]);
    }
}
</script>

```

LEGEND-LABEL.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

![[ASP.NET MVC Chart Label Color](../images/legend-label.png)]

Series Selection on Legend

By default, legend click enables you to collapse the series visibility. On other hand, if you need to select a series through legend click, disable the [ToggleVisibility](#).

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend => legend.ToggleVisibility("true")
).Title("Inflation - Consumer Price").Load("load").Render()

```

SELECTION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Enable Animation

You can customize the animation while clicking legend by setting enableAnimation as true or false using EnableAnimation property in chart.

CSHTML

```
@(Html.EJS().Chart("container").EnableAnimation(true).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr => mr.Visible(false).DataLabel(dl =>
dl.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).Name("Gold").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr => mr.Visible(false).DataLabel(dl =>
dl.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource1).Name("Silver").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
        Marker(mr => mr.Visible(false).DataLabel(dl =>
dl.Visible(true).Position(Syncfusion.EJ2.Charts.LabelPosition.Top))).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).Name("Bronze").Width(2).Add();
})).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGrid
Lines(mg => mg.Width(0))).
    PrimaryYAxis(py => py.MajorGridLines(ml => ml.Width(0)).MajorTickLines(mt
=> mt.Width(0)).LineStyle(ls => ls.Width(0)).LabelStyle(ls =>
ls.Color("transparent")))
    .Render()
)
```

ANIMATION.CS

```
public ActionResult Index()
{
    List<ColumnChartData> chartData1 = new List<ColumnChartData>
    {
        new ColumnChartData {x="USA", y=46},
        new ColumnChartData {x="GBR", y=27},
        new ColumnChartData {x="CHN", y=26}
    };
    List<ColumnChartData> chartData2 = new List<ColumnChartData>
    {
        new ColumnChartData {x="USA", y=37},
        new ColumnChartData {x="GBR", y=23},
        new ColumnChartData {x="CHN", y=18}
    };
    List<ColumnChartData> chartData3 = new List<ColumnChartData>
    {
        new ColumnChartData {x="USA", y=38},
        new ColumnChartData {x="GBR", y=17},
        new ColumnChartData {x="CHN", y=26}
    };
    ViewBag.dataSource = chartData1;
    ViewBag.dataSource1 = chartData2;
}
```

```

        ViewBag.dataSource2 = chartData3;
        return View();
    }
    public class ColumnChartData
    {
        public string x;
        public double y;
    }

```

Collapsing Legend Item

By default, series name will be displayed as legend. To skip the legend for a particular series, you can give empty string to the series name.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Load("load").Render()

```

COLLAPSE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    }

```

```

    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend Title

You can set title for legend using **Title** property in **LegendSettings**. You can also customize the **FontStyle**, **Size**, **FontWeight**, **Color**, **TextAlignment**, **FontFamily**, **Opacity** and **TextOverflow** of legend title. **TitlePosition** is used to set the legend position in **Top**, **Left** and **Right** position. **MaximumTitleWidth** is used to set the width of the legend title. By default, it will be **100px**.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend =>
legend.Position(Syncfusion.EJ2.Charts.LegendPosition.Bottom).Title("Inflation - Consumer Price")
).Load("load").Render()

```

TITLE.CS

```

public IActionResult Index()
{
    List chartData = new List
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
    }
}

```

```

        new LineChartData { xValue = new DateTime(2009, 01, 01),
yValue = 54, yValue1 = 66 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 57, yValue1 = 78 },
        new LineChartData { xValue = new DateTime(2011, 01, 01),
yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

Arrow Page Navigation

By default, the page number will be enabled while legend paging. Now, you can disable that page number and also you can get left and right arrows for page navigation. You have to set `false` value to `EnablePages` to get this support.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend =>
legend.Position(Syncfusion.EJ2.Charts.LegendPosition.Bottom).EnablePages(false)
).Title("Inflation - Consumer Price").Load("load").Render()

```

ARROW-PAGE.CS

```

public IActionResult Index()
{
    List chartData = new List
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
    }
}

```



```

        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
        new LineChartData { xValue = new DateTime(2009, 01, 01),
yValue = 54, yValue1 = 66 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 57, yValue1 = 78 },
        new LineChartData { xValue = new DateTime(2011, 01, 01),
yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

Legend Item Padding

The [ItemPadding](#) property can be used to adjust the space between the legend items.

CSHTML

```

@Html.EJS().Charts("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("England").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Russia").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional).Interval(20).Minimum(0).Maximum(100)
).LegendSettings(legend =>
legend.Position(Syncfusion.EJ2.Charts.LegendPosition.Bottom).EnablePages(false).ItemPadding(30)
).Title("Inflation - Consumer Price").Load("load").Render()

```

ITEMPADDING.CS

```

public IActionResult Index()
{
    List chartData = new List
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
    }
}

```

```

        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
        new LineChartData { xValue = new DateTime(2009, 01, 01),
yValue = 54, yValue1 = 66 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 57, yValue1 = 78 },
        new LineChartData { xValue = new DateTime(2011, 01, 01),
yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

Tooltip

<!-- markdownlint-disable MD036 -->

Chart will display details about the points through tooltip, when the mouse is moved over the point.

Enable tooltip

The tooltip is useful when you cannot display information by using the data labels due to space constraints. You can enable the tooltip by setting [Enable](#) property as true in [Tooltip](#) object.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
    PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").Tooltip(tt =>
tt.Enable(true)).Render()

```

DEFAULT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {

```

```

        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string x;
    public double yValue;
}

```

Fixed tooltip

By default, tooltip track the mouse movement, but you can set a fixed position for the tooltip by using the [Location](#) property.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("xValue").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
})).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).LegendSettings(ls => ls.Visible(false))
.Title("Olympic Medal Counts - RIO").Tooltip(tt =>
tt.Enable(true).Location(lc => lc.X(120).Y(20))).Render()

```

FIXED.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { xValue= "USA", yValue= 46 },
        new ColumnChartData { xValue= "GBR", yValue= 27 },
        new ColumnChartData { xValue= "CHN", yValue= 26 },
        new ColumnChartData { xValue= "UK", yValue= 26 },
        new ColumnChartData { xValue= "AUS", yValue= 26 },
        new ColumnChartData { xValue= "IND", yValue= 26 },
        new ColumnChartData { xValue= "DEN", yValue= 26 },
        new ColumnChartData { xValue= "MEX", yValue= 26 }
    };
}

```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class ColumnChartData
    {
        public string xValue;
        public double yValue;
    }

```

Format the tooltip

<!-- markdownlint-disable MD013 -->

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").
Tooltip(tt => tt.Enable(true).
Header("medal count").
Format("${series.name} ${point.x} : ${point.y}")).Render()

```

FORMAT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData

```

```
{
    public string x;
    public double yValue;
}
```

<!-- markdownlint-disable MD013 -->

Tooltip template

Any HTML elements can be displayed in the tooltip by using the [Template](#) property of the tooltip. You can use the \${x} and \${y} as place holders in the HTML element to display the x and y values of the corresponding data point.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").
Tooltip(tt => tt.Enable(true).
header("medal count").
template("#Medal")).Render()
<script id="Medal" type="text/x-template">
    <div id='templateWrap'>
        <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Medal</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
        </table>
    </div>
</script>
```

TEMPLATE.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
}
```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class ColumnChartData
    {
        public string x;
        public double yValue;
    }

```

Customize the appearance of tooltip

The [Fill](#) and [Border](#) properties are used to customize the background color and border of the tooltip respectively. The [TextStyle](#) property in the tooltip is used to customize the font of the tooltip text. The [HighlightColor](#) property is used to customize the point color while hovering for tooltip.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
HighlightColor("red").
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Tooltip(tt => tt.Enable(true).
Format("<b>${point.x} : ${point.y}</b>").
Fill('#7bb4eb')).Render()
}

```

CUSTOM-TOOLTIP.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string x;

```

```

        public double yValue;
    }

```

Tooltip mapping name

By default, tooltip shows information of x and y value in points. You can show more information from data source in tooltip by using the [TooltipMappingName](#) property of the tooltip. You can use the `${point.tooltip}` as place holders to display the specified tooltip content.

CSHTML

```

@ (Html.EJS () .Chart ("container") .Series (series =>
{
    series.Type (Syncfusion.EJ2.Charts.ChartSeriesType.Column) .
        XName ("xValue") .
        YName ("yValue") .
        TooltipMappingName ("text") .
        DataSource (ViewBag.dataSource) .
        Name ("Gold") .
        Width (2) .Add ();
}).
    PrimaryXAxis (px =>
px.ValueType (Syncfusion.EJ2.Charts.ValueType.Category) .Interval (1))
    .Tooltip (tp => tp.Enable (true) .Format ("${point.tooltip}"))
    .Render ()
)

```

TOOLTIP-MAPPING.CS

```

public ActionResult Index()
{
    List<GroupingChartData> chartData = new List<GroupingChartData>
    {
        new GroupingChartData { xValue = "China", yValue =
26, text = "China: 26" },
        new GroupingChartData { xValue = "Russia", yValue =
19, text = "Russia: 19" },
        new GroupingChartData { xValue = "Germany", yValue =
17, text = "Germany: 17" },
        new GroupingChartData { xValue = "Japan", yValue =
12, text = "Japan: 12" },
        new GroupingChartData { xValue = "France", yValue =
10, text = "France: 10" },
        new GroupingChartData { xValue = "South Korea", yValue =
9, text = "South Korea: 9" },
        new GroupingChartData { xValue = "Great Britain", yValue =
27, text = "Great Britain: 27" },
        new GroupingChartData { xValue = "Italy", yValue =
8, text = "Italy: 8" },
        new GroupingChartData { xValue = "Australia", yValue =
8, text = "Australia: 8" },
        new GroupingChartData { xValue = "Netherlands", yValue =
8, text = "Netherlands: 8" },
        new GroupingChartData { xValue = "Hungary", yValue =
8, text = "Hungary: 8" },
    }
}

```

```

        new GroupingChartData { xValue = "Brazil", yValue =
7, text = "Brazil: 7" },
        new GroupingChartData { xValue = "Spain", yValue =
7, text = "Spain: 7" },
        new GroupingChartData { xValue = "Kenya", yValue =
6, text = "Kenya: 6" }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class GroupingChartData
{
    public string xValue;
    public double yValue;
    public string text;
}

```

Zooming and Panning

Enable zooming

Chart can be zoomed in three ways.

- Selection - By setting [EnableSelectionZooming](#) property to true in `ZoomSettings`, you can zoom the chart by using the rubber band selection.
- Mousewheel - By setting [EnableMouseWheelZooming](#) property to true in `ZoomSettings`, you can zoom in and zoom out the chart by scrolling the mouse wheel.
- Pinch - By setting [EnablePinchZooming](#) property to true in `ZoomSettings`, you can zoom the chart through pinch gesture in touch enabled devices.

Note: Pinch zooming is supported only in browsers that support multi-touch gestures. Currently IE11, Chrome and Opera browsers support multi-touch in desktop devices.

CSHTML

```

@Html.EJS().Chart("container").
PrimaryXAxis(px => px.Title("Years").
MajorGridLines(ViewBag.majorGridLines).
ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
Skeleton("yMMM")).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
    Border(ViewBag.border).Animation(ViewBag.animation).
    DataSource("series1").
    Name("Product X").
    XName("x").
    YName("y").Add();
})
).ChartArea(area => area.Border(ViewBag.chartBorder)).
LegendSettings(legend => legend.Visible(false)).
PrimaryYAxis(py => py.Title("Profit ($)").
LineStyle(ViewBag.lineStyle).
MajorTickLines(ViewBag.majorTickLines).
RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)).
ZoomSettings(z => z.EnableMouseWheelZooming(true).

```



```

EnablePinchZooming(true).EnableSelectionZooming(true).
Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).
Title("Sales History of Product X").Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > 0.5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

DEFAULT.CS

```

public IActionResult Index()
{
    return View();
}

```

After zooming the chart, a zooming toolbar will appear with **zoom**, **zoomin**, **zoomout**, **pan** and **reset** buttons. Selecting the Pan option will allow to pan the chart and selecting the Reset option will reset the zoomed chart.

Modes

The [Mode](#) property in zoomSettings specifies whether the chart is allowed to scale along the horizontal axis or vertical axis. The default value of the mode is XY (both axis).

There are three types of mode.

- X - Allows us to zoom the chart horizontally.
- Y - Allows us to zoom the chart vertically.
- XY - Allows us to zoom the chart both vertically and horizontally.

CSHTML

```

@Html.EJS().Chart("container").
PrimaryXAxis(px => px.Title("Years").
MajorGridLines(ViewBag.majorGridLines).
ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
Skeleton("yMMM")).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
Border(ViewBag.border).Animation(ViewBag.animation).
DataSource("series1").
Name("Product X").
XName("x").

```

```

        YName("y").Add();
    }
).ChartArea(area => area.Border(ViewBag.chartBorder)).
LegendSettings(legend => legend.Visible(false)).
PrimaryYAxis(py => py.Title("Profit ($)").
LineStyle(ViewBag.lineStyle).
MajorTickLines(ViewBag.majorTickLines).
RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)).
ZoomSettings(z => z.EnableMouseWheelZooming(true).
EnablePinchZooming(true).EnableSelectionZooming(true).
Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).
Title("Sales History of Product X").Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > 0.5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

MODE.CS

```

public IActionResult Index()
{
    return View();
}

```

Toolbar

By default, zoomin, zoomout, pan and reset buttons will be displayed for zoomed chart. You can customize to show the desired options in the toolbar using the [ToolbarItems](#) property. Also using the [ShowToolbar](#) property, you can show toolkit for zooming and panning the chart during initial rendering itself.

CSHTML

```

@Html.EJS().Chart("container").
PrimaryXAxis(px => px.Title("Years").
MajorGridLines(ViewBag.majorGridLines).
ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
Skeleton("yMMM")).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
Border(ViewBag.border).Animation(ViewBag.animation).
DataSource("series1").
Name("Product X").
XName("x").

```

```

        YName("y").Add();
    }
).ChartArea(area => area.Border(ViewBag.chartBorder)).
LegendSettings(legend => legend.Visible(false)).
PrimaryYAxis(py => py.Title("Profit ($)").
LineStyle(ViewBag.lineStyle).
MajorTickLines(ViewBag.majorTickLines).
RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)).
ZoomSettings(z => z.EnableMouseWheelZooming(true).
ToolBarItems(ViewBag.toolbarItems).
ShowToolBar(true).
Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).
Title("Sales History of Product X").Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > 0.5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

TOOLBAR.CS

```

public IActionResult Index()
{
    ViewBag.toolBarItems = new String[] { "Zoom", "Pan", "Reset" };
    return View();
}

```

Enable pan

Using [EnablePan](#) property you can able to pan the zoomed chart without help of toolbar items.

CSHTML

```

@Html.EJS().Chart("container").
PrimaryXAxis(px => px.Title("Years").
MajorGridLines(ViewBag.majorGridLines).
ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
Skeleton("yMMM").ZoomFactor(0.2).ZoomPosition(0.6)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
Border(ViewBag.border).Animation(ViewBag.animation).
DataSource("series1").
Name("Product X").
XName("x").
YName("y").Add();
}

```

```

    ).ChartArea(area => area.Border(ViewBag.chartBorder)).
    LegendSettings(legend => legend.Visible(false)).
    PrimaryYAxis(py => py.Title("Profit ($)").
    LineStyle(ViewBag.lineStyle).
    MajorTickLines(ViewBag.majorTickLines).
    RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)).
    ZoomSettings(z => z.EnableMouseWheelZooming(true).
    EnablePinchZooming(true).EnableSelectionZooming(true).
    Mode(Syncfusion.EJ2.Charts.ZoomMode.X).EnablePan(true)).
    Title("Sales History of Product X").Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > 0.5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

PAN.CS

```

public IActionResult Index()
{
    return View();
}

```

Enable scrollbar

Using the [EnableScrollbar](#) property, you can add a scrollbar to a zoomed chart. This scrollbar allows you to zoom or pan the chart. The appearance of the scrollbar can be customized using properties in [ScrollbarSettings](#). For example, you can use [TrackColor](#) and [TrackRadius](#) properties to customize the track of the scrollbar, and [ScrollbarRadius](#) and [ScrollbarColor](#) properties to customize the scroller. The ability to zoom through the scrollbar can be enabled or disabled using the [EnableZoom](#) property in [ScrollbarSettings](#). Additionally, you can change the color of the grip and height of the scrollbar using the [GripColor](#) and [Height](#) properties.

CSHTML

```

@Html.EJS().Chart("container").
    PrimaryXAxis(px => px.ScrollbarSettings(scroll =>
    scroll.Enable(true).Height(14).EnableZoom(false).TrackRadius(8).ScrollbarRad
    ius(8).GripColor("transparent").TrackColor("yellow").ScrollbarColor("red")).
    ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)).Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
        Border(br => br.Width(0.5).Color("#00bdae")).Animation(an =>
        an.Enable(false)).

```

```

        DataSource("series1").
        Name("Product X").
        XName("x").
        YName("y").Add();
    }
).
LegendSettings(legend => legend.Visible(false)).
ZoomSettings(z => z.EnableScrollbar(true)
EnableSelectionZooming(true).
Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).
Title("Sales History of Product X").Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > 0.5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

SCROLLBAR.CS

```

public IActionResult Index()
{
    return View();
}

```

Auto interval on zooming

By using [EnableAutoIntervalOnZooming](#) property, the axis interval will get calculated automatically with respect to the zoomed range.

CSHTML

```

@Html.EJS().Chart("container").
    PrimaryXAxis(px => px.Title("Years").
    MajorGridLines(ViewBag.majorGridLines).
    ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
    Skeleton("yMMM")).Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).
        Border(ViewBag.border).Animation(ViewBag.animation).
        DataSource("series1").
        Name("Product X").
        XName("x").
        YName("y").Add();
    }
).ChartArea(area => area.Border(ViewBag.chartBorder)).

```

```

LegendSettings(legend => legend.Visible(false)).
enableAutoIntervalOnZooming(true).
PrimaryYAxis(py => py.Title("Profit ($)").
LineStyle(ViewBag.lineStyle).
MajorTickLines(ViewBag.majorTickLines).
RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)).
ZoomSettings(z => z.EnableSelectionZooming(true).
Mode(Syncfusion.EJ2.Charts.ZoomMode.X)).
Title("Sales History of Product X").Render()
<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 500; i++) {
        if (Math.random() > 0.5) {
            value += Math.random();
        }
        else {
            value -= Math.random();
        }
        point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

AUTO-INTERVAL.CS

```

public IActionResult Index()
{
    return View();
}

```

Data Editing

Enable Data Editing

It provides drag and drop support to the rendered points. Now, we can change the location or value of the point based on its **y** value. To enable the data editing, set the **Enable** property to true in the drag settings of the series. Also, we can set color using **Fill** property and set the data editing minimum and maximum range using **MinY** and **MaxY** properties.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .Fill("orange")
    .Marker(mr => mr.Visible(true).Width(10).Height(10))

    .XName("xValue").YName("y").DataSource(ViewBag.dataSource).Name("Product
A").Width(2).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
    .Fill("darkblue")
    .Marker(mr => mr.Visible(true).Width(10).Height(10))
}

```

```
.XName("xValue").YName("y1").DataSource(ViewBag.dataSource).Name("Product
B").Width(2).Add();
    }).PrimaryYAxis(px =>
px.Minimum(0).Maximum(100).Interval(20).LineStyle(ls =>
ls.Width(0)).MajorTickLines(mt =>
mt.Width(0)).LabelFormat("{value}%").Title("Sales")
    ).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg =>
mg.Width(0)).LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks).Mi
nimum("-0.5").Maximum("6.5")
    ).Tooltip(tr => tr.Enable(true)
    ).ChartArea(area => area.Border(br=>br.Color("transparent"))
    ).Tooltip(tt => tt.Enable(true)
    ).LegendSettings(lg => lg.Visible(false)
    ).Title("Sales Prediction of Products").Load("load").Render()
```

DEFAULT.CS

```
public IActionResult DataEditing()
{
    List<DataEditingData> chartData = new List<DataEditingData>
    {
        new DataEditingData { xValue = "2005", y = 21, y1= 21},
        new DataEditingData { xValue = "2006", y = 60, y1= 22},
        new DataEditingData { xValue = "2007", y = 45, y1= 36 },
        new DataEditingData { xValue = "2008", y = 50, y1= 34},
        new DataEditingData { xValue = "2009", y = 74, y1= 54 },
        new DataEditingData { xValue = "2010", y = 65, y1= 55},
        new DataEditingData { xValue = "2011", y = 85, y1= 60}
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class DataEditingData
{
    public string xValue;
    public double y;
    public double y1;
}
```

Crosshair

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using [Enable](#) property in the **Crosshair**.

CSHTML

```
@Html.EJS().Chart("container").
ChartArea(area => area.Border(ViewBag.ChartBorder))
.Series(series =>
{
```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    .PrimaryYAxis(py => py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .Interval(20).Minimum(0).Maximum(100)
    .Title("Inflation - Consumer Price")
    .Tooltip(tt => tt.Enable(true))
    .Crosshair(cr=>cr.Enable(true)).Render()

```

CROSSHAIR.CS

```

public ActionResult Index()
{
    return View();
}

```

Tooltip for axis

Tooltip label for an axis can be enabled by using [Enable](#) property of **CrosshairTooltip** in the corresponding axis.

CSHTML

```

@Html.EJS().Chart("container").
ChartArea(area => area.Border(ViewBag.ChartBorder))
.Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    .CrosshairTooltip(ViewBag.cross)
    .PrimaryYAxis(py => py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .Interval(20).Minimum(0).Maximum(100)
    .CrosshairTooltip(ViewBag.cross)
    .Title("Inflation - Consumer Price")
    .Tooltip(tt => tt.Enable(true))
    .Crosshair(cr=>cr.Enable(true)).Render()

```


AXIS-TOOLTIP.CS

```
public ActionResult Index()
{
    return View();
}
```

Customization

The [Fill](#) and [TextStyle](#) property of the **CrosshairTooltip** is used to customize the background color and font style of the crosshair label respectively. Color and width of the crosshair line can be customized by using the [Line](#) property in the crosshair.

CSHTML

```
@Html.EJS().Chart("container").
    ChartArea(area => area.Border(ViewBag.ChartBorder))
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();
    })
    .PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    .CrosshairTooltip(ViewBag.cross))
    .PrimaryYAxis(py => py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .Interval(20).Minimum(0).Maximum(100)
    .CrosshairTooltip(ViewBag.cross))
    .Title("Inflation - Consumer Price")
    .Tooltip(tt => tt.Enable(true))
    .Crosshair(ViewBag.CrossLine).Render()
```

CUSTOM.CS

```
public ActionResult Index()
{
    return View();
}
```

Trackball

Trackball is used to track a data point closest to the mouse or touch position. Trackball marker indicates the closest point and trackball tooltip displays the information about the point.

Trackball can be enabled by setting the [Enable](#) property of the crosshair to true and [Shared](#) property in **Tooltip** to true in chart.

CSHTML

```

@Html.EJS().Chart("container-vertical").Series(series =>
{
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("yV
alue").Marker(ViewBag.marker)
.DataSource(ViewBag.dataSource).Name("John").Width(2).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("yV
alue1").Marker(ViewBag.marker)
.DataSource(ViewBag.dataSource).Name("Andrew").Width(2).Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("yV
alue2").Marker(ViewBag.marker)
.DataSource(ViewBag.dataSource).Name("Thomas").Width(2).Add();})\
.PrimaryXAxis(px =>
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
.MajorGridLines(ViewBag.majorGridLines).LineStyle(ViewBag.lineStyle)
.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
.Skeleton("y"))
.PrimaryYAxis(py => py.Title("Revenue")
.MajorTickLines(ViewBag.majorTickLines)
.LineStyle(ViewBag.lineStyle)
.Minimum(10)
.Maximum(80)
.LabelFormat("{value}M"))
.ChartArea(area => area.Border(ViewBag.ChartBorder))
.Tooltip(tl =>tl.Enable(true).Shared(true))
.Crosshair(cr =>
cr.Enable(true).LineType(Syncfusion.EJ2.Charts.LineType.Vertical))
.Title("Average Sales per Person").Render()

```

TRACKBALL.CS

```

public ActionResult Index()
{
    List<TrackballChartData> chartData = new
List<TrackballChartData>
{
    new TrackballChartData { xValue = new DateTime(2000, 2, 11),
yValue = 14, yValue1 = 39, yValue2 = 60 },
    new TrackballChartData { xValue = new DateTime(2000, 9, 4),
yValue = 20, yValue1 = 30, yValue2 = 55 },
    new TrackballChartData { xValue = new DateTime(2001, 2, 11),
yValue = 25, yValue1 = 28, yValue2 = 48 },
    new TrackballChartData { xValue = new DateTime(2001, 9, 16),
yValue = 21, yValue1 = 35, yValue2 = 57 },
    new TrackballChartData { xValue = new DateTime(2002, 2, 7),
yValue = 13, yValue1 = 39, yValue2 = 62 },
    new TrackballChartData { xValue = new DateTime(2002, 9, 7),
yValue = 18, yValue1 = 41, yValue2 = 64 },
    new TrackballChartData { xValue = new DateTime(2003, 2, 11),
yValue = 24, yValue1 = 45, yValue2 = 57 },
    new TrackballChartData { xValue = new DateTime(2003, 9, 14),
yValue = 23, yValue1 = 48, yValue2 = 53 },
    new TrackballChartData { xValue = new DateTime(2004, 2, 6),
yValue = 19, yValue1 = 54, yValue2 = 63 },

```

```

        new TrackballChartData { xValue = new DateTime(2004, 9, 6),
yValue = 31, yValue1 = 55, yValue2 = 50 },
        new TrackballChartData { xValue = new DateTime(2005, 2, 11),
yValue = 39, yValue1 = 57, yValue2 = 66 },
        new TrackballChartData { xValue = new DateTime(2005, 9, 11),
yValue = 50, yValue1 = 60, yValue2 = 65 },
        new TrackballChartData { xValue = new DateTime(2006, 2, 11),
yValue = 24, yValue1 = 60, yValue2 = 79 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class TrackballChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
    public double yValue2;
}

```

Synchronized charts in ASP.NET MVC Chart Component

Tooltip synchronization

The tooltip can be synchronized across multiple charts using the **ShowTooltip** and **HideTooltip** methods. When we hover over a data point in one chart, we call the **ShowTooltip** method for the other charts to display related information in other connected charts simultaneously.

In the **ShowTooltip** method, specify the following parameters programmatically to enable tooltip for a particular chart:

- **X** - Data point x-value or x-coordinate value.
- **Y** - Data point y-value or y-coordinate value.

CSHTML

```

<div class="control-section">
    <div class="row">
        <div class="col" id="container1">
            @Html.EJS().Chart("lineContainer1").Series(series =>
            {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("USD").YName("EUR").Width(2)
                .EmptyPointSettings(es =>
es.Mode(Syncfusion.EJ2.Charts.EmptyPointMode.Drop)).DataSource("ChartPoints")
                ).Add();
                }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
                ).PrimaryYAxis(px =>
px.LabelFormat("n2").Minimum(0.86).Maximum(0.96).Interval(0.025).LineStyle(ls
=> ls.Width(0)).MajorTickLines(mg => mg.Width(0))
            }

```

```

        ).Tooltip(tt =>
tt.Enable(true).Shared(true).Header("").Format("<b>€${point.y}</b> <br>
${point.x} 2023").EnableMarker(false)).ChartArea(area => area.Border(br =>
br.Width(0))

        ).TitleStyle(tl =>
tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
        ).Title("US to
EURO").Load("load").ChartMouseMove("chartMouseMove1").ChartMouseLeave("chart
MouseLeave1").ChartMouseUp("chartMouseUp1").Render()
</div>
<div class="col" id="container2">
    @Html.EJS().Chart("lineContainer2").Series(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).XName("USD").YName("
INR").Width(2).Opacity(0.6).Border(br =>
br.Width(2)).DataSource("ChartPoints").Add();
        }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
        ).PrimaryYAxis(py =>
py.LabelFormat("n1").Minimum(79).Maximum(85).Interval(1.5).LineStyle(ls =>
ls.Width(0)).MajorTickLines(mg => mg.Width(0))
        ).Tooltip(tt =>
tt.Enable(true).Shared(true).Header("").Format("<b>₹${point.y}</b> <br>
${point.x} 2023").EnableMarker(false)).ChartArea(area => area.Border(br =>
br.Width(0))

        ).TitleStyle(tl =>
tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
        ).Title("US to
INR").Load("load").ChartMouseMove("chartMouseMove2").ChartMouseLeave("chartM
ouseLeave2").ChartMouseUp("chartMouseUp2").Render()
    }
</div>
</div>
<script src="chart/user-interaction/synchronization/crosshair/sync-
data.js"></script>
<script>
    var chart1;
    var chart2;
    var ChartPoints = synchronizedData;
    function load(args) {
        var selectedTheme = location.hash.split('/')[1];
        selectedTheme = selectedTheme ? selectedTheme : 'Material';
        args.chart.theme = (selectedTheme.charAt(0).toUpperCase() +
selectedTheme.slice(1)).replace(/-dark/i, 'Dark').replace(/contrast/i,
'Contrast');
        args.chart.series[0].dataSource = ChartPoints;
        args.chart.primaryXAxis.interval = ej.base.Browser.isDevice ? 2 : 1;
        args.chart.primaryXAxis.edgeLabelPlacement =
ej.base.Browser.isDevice ? 'None' : 'Shift';
        args.chart.primaryXAxis.labelRotation = ej.base.Browser.isDevice ? -
45 : 0;
        args.chart.tooltip.fadeOutDuration = ej.base.Browser.isDevice ? 2500
: 1000;
    }

```

```

function chartMouseMove1(args) {
    chart1 = document.getElementById('lineContainer1').ej2_instances[0];
    chart2 = document.getElementById('lineContainer2').ej2_instances[0];
    if ((!ej.base.Browser.isDevice && !chart1.isTouch &&
!chart1.isChartDrag) || chart1.startMove) {
        chart2.startMove = chart1.startMove;
        chart2.showTooltip(args.x, args.y);
    }
}
function chartMouseMove2(args) {
    chart1 = document.getElementById('lineContainer1').ej2_instances[0];
    chart2 = document.getElementById('lineContainer2').ej2_instances[0];
    if ((!ej.base.Browser.isDevice && !chart2.isTouch &&
!chart2.isChartDrag) || chart2.startMove) {
        chart1.startMove = chart2.startMove;
        chart1.showTooltip(args.x, args.y);
    }
}
function chartMouseLeave1(args) {
    chart2 = document.getElementById('lineContainer2').ej2_instances[0];
    chart2.hideTooltip();
}
function chartMouseLeave2(args) {
    chart1 = document.getElementById('lineContainer1').ej2_instances[0];
    chart1.hideTooltip();
}
function chartMouseUp1(args) {
    chart1 = document.getElementById('lineContainer1').ej2_instances[0];
    chart2 = document.getElementById('lineContainer2').ej2_instances[0];
    if (ej.base.Browser.isDevice && chart1.startMove) {
        chart2.hideTooltip();
    }
}
function chartMouseUp2(args) {
    chart1 = document.getElementById('lineContainer1').ej2_instances[0];
    chart2 = document.getElementById('lineContainer2').ej2_instances[0];
    if (ej.base.Browser.isDevice && chart2.startMove) {
        chart1.hideTooltip();
    }
}
</script>
<style>
    #control-container {
        padding: 0px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
</style>

```

TOOLTIP.CS

```
public IActionResult Index()
{
    return View();
}
```

Crosshair synchronization

The crosshair can be synchronized across multiple charts using the **ShowCrosshair** and **HideCrosshair** methods. When we hover over one chart, we call the **ShowCrosshair** method for the other charts to align with data points in other connected charts, simplifying data comparison and analysis.

In the **ShowCrosshair** method, specify the following parameters programmatically to enable crosshair for a particular chart:

- **X** - Specifies the x-value of the point or x-coordinate.
- **Y** - Specifies the y-value of the point or y-coordinate.

CSHTML

```
<div class="control-section">
    <div class="row">
        <div class="col" id="container1">
            @Html.EJS().Chart("lineContainer1").Series(series =>
            {
                series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).XName("USD").YName("EUR").Width(2)
                    .EmptyPointSettings(es =>
                    es.Mode(Syncfusion.EJ2.Charts.EmptyPointMode.Drop)).DataSource("ChartPoints")
                    .Add();
                }).PrimaryXAxis(px =>
                px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
                ).PrimaryYAxis(px =>
                px.LabelFormat("n2").Minimum(0.86).Maximum(0.96).Interval(0.025).LineStyle(ls => ls.Width(0)).MajorTickLines(mg => mg.Width(0))
                ).ChartArea(area => area.Border(br => br.Width(0))
                ).TitleStyle(tl =>
                tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
                ).Crosshair(cr =>
                cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true).DashArray("2,2")
                ).Title("US to EURO").Load("load").ChartMouseMove("chartMouseMove1").ChartMouseLeave("chartMouseLeave1").ChartMouseUp("chartMouseUp1").Render()
            }
        </div>
        <div class="col" id="container2">
            @Html.EJS().Chart("lineContainer2").Series(series =>
            {
                series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Area).XName("USD").YName("INR").Width(2).Opacity(0.6).Border(br => br.Width(2)).DataSource("ChartPoints").Add();
            }
```

```

        }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
        ).PrimaryYAxis(px =>
px.LabelFormat("n1").Minimum(79).Maximum(85).Interval(1.5).LineStyle(ls =>
ls.Width(0)).MajorTickLines(mg => mg.Width(0))
        ).ChartArea(area => area.Border(br => br.Width(0))
        ).TitleStyle(tl =>
tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
        ).Crosshair(cr =>
cr.LineType(Syncfusion.EJ2.Charts.LineType.Vertical).Enable(true).DashArray(
"2,2")
        ).Title("US to
INR").Load("load").ChartMouseMove("chartMouseMove2").ChartMouseLeave("chartM
ouseLeave2").ChartMouseUp("chartMouseUp2").Render()
    </div>
</div>
</div>
<script src="chart/user-interaction/synchronization/crosshair/sync-
data.js"></script>
<script>
    var chart1;
    var chart2;
    var ChartPoints = synchronizedData;
    function load(args) {
        var selectedTheme = location.hash.split('/')[1];
        selectedTheme = selectedTheme ? selectedTheme : 'Material';
        args.chart.theme = (selectedTheme.charAt(0).toUpperCase() +
selectedTheme.slice(1)).replace(/-dark/i, 'Dark').replace(/contrast/i,
'Contrast');
        args.chart.series[0].dataSource = ChartPoints;
        args.chart.primaryXAxis.interval = ej.base.Browser.isDevice ? 2 : 1;
        args.chart.primaryXAxis.edgeLabelPlacement =
ej.base.Browser.isDevice ? 'None' : 'Shift';
        args.chart.primaryXAxis.labelRotation = ej.base.Browser.isDevice ? -
45 : 0;
    }
    function chartMouseMove1(args) {
        chart1 = document.getElementById('lineContainer1').ej2_instances[0];
        chart2 = document.getElementById('lineContainer2').ej2_instances[0];
        if ((!ej.base.Browser.isDevice && !chart1.isTouch &&
!chart1.isChartDrag) || chart1.startMove) {
            chart2.startMove = chart1.startMove;
            chart2.showCrosshair(args.x, args.y);
        }
    }
    function chartMouseMove2(args) {
        chart1 = document.getElementById('lineContainer1').ej2_instances[0];
        chart2 = document.getElementById('lineContainer2').ej2_instances[0];
        if ((!ej.base.Browser.isDevice && !chart2.isTouch &&
!chart2.isChartDrag) || chart2.startMove) {
            chart1.startMove = chart2.startMove;
            chart1.showCrosshair(args.x, args.y);
        }
    }
    function chartMouseLeave1(args) {

```

```

        chart2 = document.getElementById('lineContainer2').ej2_instances[0];
        chart2.hideCrosshair();
    }
    function chartMouseLeave2(args) {
        chart1 = document.getElementById('lineContainer1').ej2_instances[0];
        chart1.hideCrosshair();
    }
    function chartMouseUp1(args) {
        chart1 = document.getElementById('lineContainer1').ej2_instances[0];
        chart2 = document.getElementById('lineContainer2').ej2_instances[0];
        if (ej.base.Browser.isDevice && chart1.startMove) {
            chart2.hideCrosshair();
        }
    }
    function chartMouseUp2(args) {
        chart1 = document.getElementById('lineContainer1').ej2_instances[0];
        chart2 = document.getElementById('lineContainer2').ej2_instances[0];
        if (ej.base.Browser.isDevice && chart2.startMove) {
            chart1.hideCrosshair();
        }
    }
</script>
<style>
    #control-container {
        padding: 0px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
</style>

```

CROSSHAIR.CS

```

public IActionResult Index()
{
    return View();
}

```

Zooming synchronization

You can maintain constant zoom levels across multiple charts using the [ZoomComplete](#) event. In the [ZoomComplete](#) event, obtain the [ZoomFactor](#) and [ZoomPosition](#) values of the particular chart, and then apply those values to the other charts.

CSHTML

```

<div class="control-section">
    <div class="row">
        <div class="col" id="container1">
            @Html.EJS().Chart("lineContainer1").Series(series =>
            {

```



```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("USD").YName("EUR").Width(2)
        .EmptyPointSettings(es =>
es.Mode(Syncfusion.EJ2.Charts.EmptyPointMode.Drop)).DataSource("ChartPoints")
).Add();
        ))).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
        ).PrimaryYAxis(px =>
px.LabelFormat("n2").Minimum(0.86).Maximum(0.96).Interval(0.025).LineStyle(ls => ls.Width(0)).MajorTickLines(mg => mg.Width(0))
        ).ChartArea(area => area.Border(br => br.Width(0))
        ).TitleStyle(tl =>
tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
        ).ZoomSettings(zn =>
zn.EnableMouseWheelZooming(true).EnableScrollbar(false).EnableDeferredZoomin
g(false).EnableSelectionZooming(true).EnablePan(true).Mode(Syncfusion.EJ2.Ch
arts.ZoomMode.X).EnablePinchZooming(true).ToolBarItems(ViewBag.toolbarItems)
        ).Title("US to
EURO").Load("load").ZoomComplete("zoomComplete").Render()
</div>
<div class="col" id="container2">
    @Html.EJS().Chart("lineContainer2").Series(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.SplineArea).XName("USD").Y
Name("INR").Width(2).Opacity(0.6).Border(br =>
br.Width(2)).DataSource("ChartPoints").Add();
        ))).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
        ).PrimaryYAxis(px =>
px.LabelFormat("n1").Minimum(79).Maximum(85).Interval(1.5).LineStyle(ls =>
ls.Width(0)).MajorTickLines(mg => mg.Width(0))
        ).ChartArea(area => area.Border(br => br.Width(0))
        ).TitleStyle(tl =>
tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
        ).ZoomSettings(zn =>
zn.EnableMouseWheelZooming(true).EnableScrollbar(false).EnableDeferredZoomin
g(false).EnableSelectionZooming(true).EnablePan(true).Mode(Syncfusion.EJ2.Ch
arts.ZoomMode.X).EnablePinchZooming(true).ToolBarItems(ViewBag.toolbarItems)
        ).Title("US to
INR").Load("load").ZoomComplete("zoomComplete").Render()
</div>
</div>
</div>
<script src="chart/user-interaction/synchronization/crosshair/sync-
data.js"></script>
<script>
    var chart1;
    var chart2;
    var charts = [];
    var zoomFactor = 0;
    var zoomPosition = 0;

```

```

var ChartPoints = synchronizedData;
function load(args) {
    var selectedTheme = location.hash.split('/')[1];
    selectedTheme = selectedTheme ? selectedTheme : 'Material';
    args.chart.theme = (selectedTheme.charAt(0).toUpperCase() +
selectedTheme.slice(1).replace(/-dark/i, 'Dark').replace(/contrast/i,
'Contrast'));
    args.chart.series[0].dataSource = ChartPoints;
    args.chart.primaryXAxis.interval = ej.base.Browser.isDevice ? 2 : 1;
    args.chart.primaryXAxis.edgeLabelPlacement =
ej.base.Browser.isDevice ? 'None' : 'Shift';
    args.chart.primaryXAxis.labelRotation = ej.base.Browser.isDevice ? -
45 : 0;
}
function zoomComplete(args) {
    if (args.axis.name === 'primaryXAxis') {
        zoomFactor = args.currentZoomFactor;
        zoomPosition = args.currentZoomPosition;
        zoomCompleteFunction(args);
    }
}
function zoomCompleteFunction(args) {
    chart1 = document.getElementById('lineContainer1').ej2_instances[0];
    chart2 = document.getElementById('lineContainer2').ej2_instances[0];
    charts = [chart1, chart2];
    for (var i = 0; i < charts.length; i++) {
        if (args.axis.series[0].chart.element.id !==
charts[i].element.id) {
            charts[i].primaryXAxis.zoomFactor = zoomFactor;
            charts[i].primaryXAxis.zoomPosition = zoomPosition;
            charts[i].zoomModule.isZoomed =
args.axis.series[0].chart.zoomModule.isZoomed;
            charts[i].zoomModule.isPanning =
args.axis.series[0].chart.zoomModule.isPanning;
        }
    }
}
</script>
<style>
#control-container {
    padding: 0px !important;
}
.row {
    display: flex;
}
.col {
    width: 50%;
    margin: 10px;
    height: 270px;
}
</style>

```

ZOOMING.CS

```

public IActionResult Index()
{

```

```

ViewBag.toolBarItems = new String[] { "Pan", "Reset" };
return View();
}

```

Selection synchronization

You can select the data across multiple charts using the [SelectionComplete](#) event. In the [SelectionComplete](#) event, obtain the selected values of the particular chart, and then apply those values to the other charts.

CSHTML

```

<div class="control-section">
  <div class="row">
    <div class="col" id="container1">
      @Html.EJS().Chart("lineContainer1").Series(series =>
      {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("USD").YName("EUR").Width(2)
          .EmptyPointSettings(es =>
            es.Mode(Syncfusion.EJ2.Charts.EmptyPointMode.Drop)).DataSource("ChartPoints")
              .Add();
          }).PrimaryXAxis(px =>
            px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
              DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
                d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
                  .PrimaryYAxis(px =>
                    px.LabelFormat("n2").Minimum(0.86).Maximum(0.96).Interval(0.025).LineStyle(ls => ls.Width(0))
                      .MajorTickLines(mg => mg.Width(0))
                        .ChartArea(area => area.Border(br => br.Width(0))
                          .TitleStyle(tl =>
                            tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)
                              .ZoomSettings(zn =>
                                zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X)
                                  .Title("US to
                                    EURO").Load("load").ZoomComplete("zoomComplete").SelectionComplete("selectio
                                      nComplete").SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Point).Selecti
                                        onPattern(Syncfusion.EJ2.Charts.SelectionPattern.Box).Render()
                                  )
                                )
                              )
                            )
                          )
                        )
                      )
                    )
                  )
                )
              )
            )
          )
        )
      }
    </div>
    <div class="col" id="container2">
      @Html.EJS().Chart("lineContainer2").Series(series =>
      {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).XName("USD").YName("INR").Width(2).Border(br => br.Width(2)).DataSource("ChartPoints").Add();
          }).PrimaryXAxis(px =>
            px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).Minimum(new
              DateTime(2023, 2, 18)).Maximum(new DateTime(2023, 8, 18)).LabelFormat("MMM
                d").MajorGridLines(mg => mg.Width(0)).LineStyle(ls => ls.Width(0))
                  .PrimaryYAxis(px =>
                    px.LabelFormat("n1").Minimum(79).Maximum(85).Interval(1.5).LineStyle(ls => ls.Width(0))
                      .MajorTickLines(mg => mg.Width(0))
                        .ChartArea(area => area.Border(br => br.Width(0))
                          .TitleStyle(tl =>
                            tl.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Near)

```

```

        ).ZoomSettings(zn =>
        zn.EnableSelectionZooming(true).Mode(Syncfusion.EJ2.Charts.ZoomMode.X)
        ).Title("US to
        INR").Load("load").ZoomComplete("zoomComplete").SelectionComplete("selection
        Complete").SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Point).Selectio
        nPattern(Syncfusion.EJ2.Charts.SelectionPattern.Box).Render()
    </div>
</div>
</div>
<script src="chart/user-interaction/synchronization/crosshair/sync-
data.js"></script>
<script>
    var chart1;
    var chart2;
    var count;
    var charts = [];
    var zoomFactor = 0;
    var zoomPosition = 0;
    var ChartPoints = synchronizedData;
    function load(args) {
        var selectedTheme = location.hash.split('/')[1];
        selectedTheme = selectedTheme ? selectedTheme : 'Material';
        args.chart.theme = (selectedTheme.charAt(0).toUpperCase() +
        selectedTheme.slice(1).replace(/-dark/i, 'Dark').replace(/contrast/i,
        'Contrast'));
        args.chart.series[0].dataSource = ChartPoints;
        args.chart.primaryXAxis.interval = ej.base.Browser.isDevice ? 2 : 1;
        args.chart.primaryXAxis.edgeLabelPlacement =
        ej.base.Browser.isDevice ? 'None' : 'Shift';
        args.chart.primaryXAxis.labelRotation = ej.base.Browser.isDevice ? -
        45 : 0;
    }
    function zoomComplete(args) {
        if (args.axis.name === 'primaryXAxis') {
            zoomFactor = args.currentZoomFactor;
            zoomPosition = args.currentZoomPosition;
            zoomCompleteFunction(args);
        }
    }
    function zoomCompleteFunction(args) {
        chart1 = document.getElementById('lineContainer1').ej2_instances[0];
        chart2 = document.getElementById('lineContainer2').ej2_instances[0];
        charts = [chart1, chart2];
        for (var i = 0; i < charts.length; i++) {
            if (args.axis.series[0].chart.element.id !==
            charts[i].element.id) {
                charts[i].primaryXAxis.zoomFactor = zoomFactor;
                charts[i].primaryXAxis.zoomPosition = zoomPosition;
                charts[i].zoomModule.isZoomed =
                args.axis.series[0].chart.zoomModule.isZoomed;
                charts[i].zoomModule.isPanning =
                args.axis.series[0].chart.zoomModule.isPanning;
            }
        }
    }
    function selectionComplete (args) {
        selectionCompleteFunction(args);
    }

```

```

    }
    function selectionCompleteFunction(args) {
        if (count == 0) {
            for (var j = 0; j < args.selectedDataValues.length; j++) {
                args.selectedDataValues[j].point =
args.selectedDataValues[j].pointIndex;
                args.selectedDataValues[j].series =
args.selectedDataValues[j].seriesIndex;
            }
            for (var i = 0; i < charts.length; i++) {
                if (args.chart.element.id !== charts[i].element.id) {
                    charts[i].selectedDataIndexes = args.selectedDataValues;
                    count += 1;
                    charts[i].dataBind();
                }
            }
            count = 0;
        }
    }
}
</script>
<style>
    #control-container {
        padding: 0px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
</style>

```

SELECTION.CS

```

public IActionResult Index()
{
    return View();
}

```

<!-- markdownlint-disable MD036 -->

Selection in ASP.NET MVC Chart Component

Chart provides selection support for the series and its data points on mouse click.

Note: When Mouse is clicked on the data points, the corresponding series legend will also be selected.

We have different type of selection mode for selecting the data. They are,

- None
- Point
- Series
- Cluster

- DragXY
- DragX
- DragY

Point

You can select a point, by setting **SelectionMode** to point.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").Add();
}).PrimaryXAxis(px => px.Title("Countries").Interval(1).

LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).
ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
PrimaryYAxis(py => py.LabelFormat("{value}%").
Title("Distribution").
Interval(20)).
Title("Age Distribution by Country").
SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Point).Render()
)
```

POINT-SELECTION.CS

```
public IActionResult Column()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
};
```

```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class ColumnChartData
    {
        public string country;
        public double gold;
        public double silver;
        public double bronze;
    }

```

Series

You can select a series, by setting **SelectionMode** to series.

CSHTML

```

@ (Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").Add();
}).PrimaryXAxis(px => px.Title("Countries").Interval(1)).

LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).
ValueType(Syncfusion.EJ2.Charts.ValueType.Category).
PrimaryYAxis(py => py.LabelFormat("{value}%").
Title("Distribution").
Interval(20)).
Title("Age Distribution by Country").
SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Series).Render()
)

```

SERIES-SELECTION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
    }
}

```

```

        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Cluster

You can select the points that corresponds to the same index in all the series, by setting **SelectionMode** to cluster.

CSHTML

```

@ (Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").Add();
}).PrimaryXAxis(px => px.Title("Countries").Interval(1)).

LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).
ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
PrimaryYAxis(py => py.LabelFormat("{value}%").
Title("Distribution").
Interval(20)).
Title("Age Distribution by Country").
SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Cluster).Render()
)

```

CLUSTER-SELECTION.CS

```

public IActionResult Column()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },

```



```

        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Rectangular selection

DragXY, DragX and DragY

To fetch the collection of data under a particular region, you have to set **SelectionMode** as **DragXY**.

- DragXY - Allows us to select data with respect to horizontal and vertical axis.
- DragX - Allows us to select data with respect to horizontal axis.
- DragY - Allows us to select data with respect to vertical axis.

The selected data's are returned as an array collection in the [DragComplete](#) event.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter).XName("xValue").Y
    Name("yValue").DataSource(ViewBag.dataSource).Marker(ViewBag.marker).Name("P
    roduct A").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Scatter).XName("xValue").Y
    Name("yValue1").DataSource(ViewBag.dataSource).Marker(ViewBag.marker1).Name(
    "Product B").Add();
}) .PrimaryXAxis(px =>
px.Title("Countries").Minimum(1970).Maximum(2016).MajorGridLines(ViewBag.maj
orGridLines)
) .PrimaryYAxis(py =>
py.LabelFormat("{value}%").Title("Sale").Interval(25).Maximum(100).Minimum(0
) .MajorGridLines(ViewBag.majorGridLines)

```

```

        ).Title("Profit Comparision of A and B").ChartArea(area =>
        area.Border(ViewBag.ChartBorder)).Load("load").SelectionMode(Syncfusion.EJ2.
        Charts.SelectionMode.DragXY).Render()
    }
}

```

DRAG.CS

```

public ActionResult Index()
{
    List<RangeSelectionChartData> chartData = new
    List<RangeSelectionChartData>
    {
        new RangeSelectionChartData { xValue = 1971, yValue = 50,
        yValue1 = 23 },
        new RangeSelectionChartData { xValue = 1972, yValue = 20,
        yValue1 = 67 },
        new RangeSelectionChartData { xValue = 1973, yValue = 63,
        yValue1 = 83 },
        new RangeSelectionChartData { xValue = 1974, yValue = 81,
        yValue1 = 43 },
        new RangeSelectionChartData { xValue = 1975, yValue = 64,
        yValue1 = 8 },
        new RangeSelectionChartData { xValue = 1976, yValue = 36,
        yValue1 = 41 },
        new RangeSelectionChartData { xValue = 1977, yValue = 22,
        yValue1 = 56 },
        new RangeSelectionChartData { xValue = 1978, yValue = 78,
        yValue1 = 31 },
        new RangeSelectionChartData { xValue = 1979, yValue = 60,
        yValue1 = 29 },
        new RangeSelectionChartData { xValue = 1980, yValue = 41,
        yValue1 = 87 },
        new RangeSelectionChartData { xValue = 1981, yValue = 62,
        yValue1 = 43 },
        new RangeSelectionChartData { xValue = 1982, yValue = 56,
        yValue1 = 12 },
        new RangeSelectionChartData { xValue = 1983, yValue = 96,
        yValue1 = 38 },
        new RangeSelectionChartData { xValue = 1984, yValue = 48,
        yValue1 = 67 },
        new RangeSelectionChartData { xValue = 1985, yValue = 23,
        yValue1 = 49 },
        new RangeSelectionChartData { xValue = 1986, yValue = 54,
        yValue1 = 67 },
        new RangeSelectionChartData { xValue = 1987, yValue = 73,
        yValue1 = 83 },
        new RangeSelectionChartData { xValue = 1988, yValue = 56,
        yValue1 = 16 },
        new RangeSelectionChartData { xValue = 1989, yValue = 67,
        yValue1 = 89 },
        new RangeSelectionChartData { xValue = 1990, yValue = 79,
        yValue1 = 18 },
        new RangeSelectionChartData { xValue = 1991, yValue = 18,
        yValue1 = 46 },
        new RangeSelectionChartData { xValue = 1992, yValue = 78,
        yValue1 = 39 },
    }
}

```

```

yValue1 = 68 },
    new RangeSelectionChartData { xValue = 1994, yValue = 43,
yValue1 = 87 },
    new RangeSelectionChartData { xValue = 1995, yValue = 29,
yValue1 = 45 },
    new RangeSelectionChartData { xValue = 1996, yValue = 14,
yValue1 = 42 },
    new RangeSelectionChartData { xValue = 1997, yValue = 85,
yValue1 = 28 },
    new RangeSelectionChartData { xValue = 1998, yValue = 24,
yValue1 = 82 },
    new RangeSelectionChartData { xValue = 1999, yValue = 61,
yValue1 = 13 },
    new RangeSelectionChartData { xValue = 2000, yValue = 80,
yValue1 = 83 },
    new RangeSelectionChartData { xValue = 2001, yValue = 14,
yValue1 = 26 },
    new RangeSelectionChartData { xValue = 2002, yValue = 34,
yValue1 = 57 },
    new RangeSelectionChartData { xValue = 2003, yValue = 81,
yValue1 = 48 },
    new RangeSelectionChartData { xValue = 2004, yValue = 70,
yValue1 = 84 },
    new RangeSelectionChartData { xValue = 2005, yValue = 21,
yValue1 = 64 },
    new RangeSelectionChartData { xValue = 2006, yValue = 70,
yValue1 = 24 },
    new RangeSelectionChartData { xValue = 2007, yValue = 32,
yValue1 = 82 },
    new RangeSelectionChartData { xValue = 2008, yValue = 43,
yValue1 = 37 },
    new RangeSelectionChartData { xValue = 2009, yValue = 21,
yValue1 = 68 },
    new RangeSelectionChartData { xValue = 2010, yValue = 63,
yValue1 = 37 },
    new RangeSelectionChartData { xValue = 2011, yValue = 9 ,
yValue1 = 35},
    new RangeSelectionChartData { xValue = 2012, yValue = 51,
yValue1 = 81 },
    new RangeSelectionChartData { xValue = 2013, yValue = 25,
yValue1 = 38 },
    new RangeSelectionChartData { xValue = 2014, yValue = 96,
yValue1 = 51 },
    new RangeSelectionChartData { xValue = 2015, yValue = 32,
yValue1 = 58 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class RangeSelectionChartData
{
    public double xValue;
    public double yValue;
    public double yValue1;
}
}

```

```
}
```

Selection type

You can select multiple points or series, by enabling the [IsMultiSelect](#) property.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").Add();
}).PrimaryXAxis(px => px.Title("Countries").Interval(1)).

LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).
ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
PrimaryYAxis(py => py.LabelFormat("{value}%").
Title("Distribution").
Interval(20)).
Title("Age Distribution by Country").
SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Series).
IsMultiSelect("true"),.Render()
)
```

SELECTION-TYPE.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
```

```

    }
    public class ColumnChartData
    {
        public string country;
        public double gold;
        public double silver;
        public double bronze;
    }

```

Selection on load

You can able to select a point or series programmatically on a chart using [SelectedDataIndexes](#) property.

CSHTML

```

@ (Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").SelectionMode("chartSelection1").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").SelectionMode("chartSelection2").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").SelectionMode("chartSelection3").Add();
}) .PrimaryXAxis(px => px.Title("Countries").Interval(1) .
LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90) .
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift) .
ValueType(Syncfusion.EJ2.Charts.ValueType.Category) ) .
PrimaryYAxis(py => py.LabelFormat("{value}%") .
Title("Distribution") .
Interval(20)) .
Title("Age Distribution by Country") .
SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Series) .Render()
)
<style>
.chartSelection1 {
    fill: red
}
.chartSelection2 {

    fill: green
}
.chartSelection3 {
    fill: blue
}
</style>

```

ONLOAD.CS

```

public IActionResult Index()

```

```

    {
        List<ColumnChartData> chartData = new List<ColumnChartData>
        {
            new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
            new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
            new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
            new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
            new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
            new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
            new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
            new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
        };
        ViewBag.dataSource = chartData;
        return View();
    }
    public class ColumnChartData
    {
        public string country;
        public double gold;
        public double silver;
        public double bronze;
    }

```

Selection through on legend

You can able to select a point or series through on legend using [ToggleVisibility](#) property. Also, use [EnableHighlight](#) property for highlighting the series through legend.

CSHTML

```

@ (Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").Add();
}) .PrimaryXAxis(px => px.Title("Countries").Interval(1) .

LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90) .
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift) .
ValueType(Syncfusion.EJ2.Charts.ValueType.Category) ) .
PrimaryYAxis(py => py.LabelFormat("{value}%") .
Title("Distribution") .
Interval(20) ) .

```

```

        Title("Age Distribution by Country").
        LegendSettings(leg => leg.Visible(true)
            .ToggleVisibility(false).EnabelHighlight(true)).
        SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Series).Render()
    )

```

SELECTION-LEGEND.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
        bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
        bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
        bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
        56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Customization for selection

You can apply custom style to selected points or series with [SelectionStyle](#) property.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue").DataSource(ViewBag.dataSource).Name("Age 0-14").SelectionStyle("chartSelection1").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue1").DataSource(ViewBag.dataSource).Name("Age 15-64").SelectionStyle("chartSelection2").Add();
}
)

```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("xValue").YName("yValue2").DataSource(ViewBag.dataSource).Name("Age 65 & Above").SelectionMode("chartSelection3").Add();
    }).PrimaryXAxis(px => px.Title("Countries").Interval(1)).

LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90).
    EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).
    ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    PrimaryYAxis(py => py.LabelFormat("{value}%").
    Title("Distribution").
    Interval(20)).
    Title("Age Distribution by Country").
    SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Series).Render()
)
<style>
.chartSelection1 {
    fill: red
}
.chartSelection2 {

    fill: green

}
.chartSelection3 {
    fill: blue
}
</style>

```

CUSTOM-SELECTION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
        new ColumnChartData{ country= "France", gold=50, silver= 45,
bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
bronze=27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData

```



```

{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Print and Export

Print

The rendered chart can be printed directly from the browser by calling the public method `print`. You can pass array of ID of elements or element to this method. By default it take element of the chart.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
})
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
    )
    .Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

PRINT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

```

```

    }
    public class ColumnChartData
    {
        public string x;
        public double yValue;
    }

```

Export

The rendered chart can be exported to **JPEG**, **PNG**, **SVG**, **PDF**, **XLSX**, or **CSV** format using the export method in chart. The input parameters for this method are **Type** for format and **FileName** for result.

The optional parameters for this method are,

- **Orientation** - either portrait or landscape mode during PDF export,
- **Controls** - pass collections of controls for multiple export,
- **Width** - width of chart export,
- **Height** - height of chart export,
- **Header** - header for the exported chart, and
- **Footer** - footer for the exported chart.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    ).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()

```

EXPORT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
    }

```

```

        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string x;
    public double yValue;
}

```

Adding header and footer in PDF export

In the export method, specify the following parameters to add a header and footer text to the exported PDF document:

- **Header** - Specify the text that should appear at the top of the exported PDF document.
- **Footer** - Specify the text that should appear at the bottom of the exported PDF document.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("x")
    .YName("y").DataSource(ViewBag.dataSource).Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Manager").Major
rGridLines(mg=>mg.Width(0))
).PrimaryYAxis(py =>
py.Title("Sales").MajorTickLines(mt=>mt.Width(0)).Minimum(0).Maximum(20000)
).Title("Sales Comparision").Render()
<div>

@Html.EJS().Button("togglebtn").IsPrimary(true).Content("Export").Render()
</div>
<script>
    document.getElementById('togglebtn').onclick = () => {
        var chart = document.getElementById('container').ej2_instances[0];
        const header = {
            content: 'Chart Header',
            fontSize: 15
        };
        const footer = {
            content: 'Chart Footer',
            fontSize: 15,
        };
        chart.exportModule.export('PDF', 'Chart', 1, [chart], null, null,
true, header, footer);
    };
}

```

```
};
</script>
```

HEADER-FOOTER.CS

```
public IActionResult Index()
{
    List<ExcelExportChartData> chartData = new List<ExcelExportChartData>
    {
        new ExcelExportChartData { x= "John", y= 10000 },
        new ExcelExportChartData { x= "Jake", y= 12000 },
        new ExcelExportChartData { x= "Peter", y= 18000 },
        new ExcelExportChartData { x= "James", y= 11000 },
        new ExcelExportChartData { x= "Mary", y= 9700 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ExcelExportChartData
{
    public string x;
    public double y;
}
```

Exporting charts into separate page during the PDF export

During PDF export, set the `ExportToMultiplePage` parameter to **true** to export each chart as a separate page.

CSHTML

```
@Html.EJS().Chart("container1").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("x").
    YName("y").DataSource(ViewBag.dataSource).Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(mg=>mg
.Width(0))
).PrimaryYAxis(py => py.Title("Million Metric
Tons").MajorTickLines(mt=>mt.Width(0)).Minimum(0).Maximum(20).Interval(4)
).Title("Crude Steel Production Annual Growth").Render()
@Html.EJS().Chart("container2").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2).XName("x"
).YName("y1").DataSource(ViewBag.dataSource).Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(mg=>mg
.Width(0))
).PrimaryYAxis(py => py.Title("Million Metric
Tons").MajorTickLines(mt=>mt.Width(0)).Minimum(0).Maximum(20).Interval(4)
).Title("Steel Production").Render()
@Html.EJS().Chart("container3").Series(series =>
{
```

```

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).Width(2).XName("x")
).YName("y2").DataSource(ViewBag.dataSource).Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(mg=>mg
.Width(0))
).PrimaryYAxis(py => py.Title("Million Metric
Tons").MajorTickLines(mt=>mt.Width(0)).Minimum(0).Maximum(20).Interval(4)
).Title("Annual Growth of Crude Steel Production").Render()
<div>
    @Html.EJS().Button("togglebtn").IsPrimary(true).Content("Export").Render()
</div>
<script>
    document.getElementById('togglebtn').onclick = () => {
        var chart1 = document.getElementById('container1').ej2_instances[0];
        var chart2 = document.getElementById('container2').ej2_instances[0];
        var chart3 = document.getElementById('container3').ej2_instances[0];
        chart1.exportModule.export('PDF', 'Chart', null, [chart1, chart2,
chart3], null, null, true, undefined, undefined, true);
    };
</script>

```

MULTI-PAGE.CS

```

public IActionResult Index()
{
    List<MultiExportChartData> chartData = new List<MultiExportChartData>
    {
        new MultiExportChartData { x = new DateTime(2012, 01, 01), y = 11.0,
y1 = 19.5, y2 = 7.1 },
        new MultiExportChartData { x = new DateTime(2013, 01, 01), y = 12.9,
y1 = 17.5, y2 = 6.8 },
        new MultiExportChartData { x = new DateTime(2014, 01, 01), y = 13.4,
y1 = 15.5, y2 = 4.1 },
        new MultiExportChartData { x = new DateTime(2015, 01, 01), y = 13.7,
y1 = 10.3, y2 = 2.8 },
        new MultiExportChartData { x = new DateTime(2016, 01, 01), y = 12.7,
y1 = 7.8, y2 = 2.8 },
        new MultiExportChartData { x = new DateTime(2017, 01, 01), y = 12.5,
y1 = 5.7, y2 = 3.8 },
        new MultiExportChartData { x = new DateTime(2018, 01, 01), y = 12.7,
y1 = 5.9, y2 = 4.3 },
        new MultiExportChartData { x = new DateTime(2019, 01, 01), y = 12.4,
y1 = 5.6, y2 = 4.7 },
        new MultiExportChartData { x = new DateTime(2020, 01, 01), y = 13.5,
y1 = 5.3, y2 = 5.6 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class MultiExportChartData
{
    public DateTime x;
    public double y;
    public double y1;
    public double y2;
}

```

```
}
```

```
<!-- markdownlint-disable MD036 -->
```

Export with optional parameters

```
<!-- markdownlint-disable MD036 -->
```

The rendered chart can be exported to specific orientation, width and height by passing it as optional parameters in the export method of chart

CSHTML

```
@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").Marker(ViewBag.marker).YName("yValue").DataSource(ViewBag.dataSource).Name("Germany").Add();

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue").YName("yValue1").Marker(ViewBag.marker).DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(ViewBag.majorGridLines).MajorGridLines(ViewBag.majorGridLines).IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years).EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
).PrimaryYAxis(py =>
py.LabelFormat("{value}%").RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(ViewBag.majorTickLines).MinorTickLines(ViewBag.minorTickLines).LineStyle(ViewBag.lineStyle).Interval(20).Minimum(0).Maximum(100)
)

).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Load("load").Render()
```

EXPORT.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
}
```

```

        public string x;
        public double yValue;
    }

```

Multiple chart export

You can export the multiple charts in single page by passing the multiple chart objects in the export method of chart. To export multiple charts in a single page, follow the given steps:

Initially, render more than one chart to export, and then add button to export the multiple charts. In button click, call the export method in charts, and then pass the multiple chart objects in the export method.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2)

    .XName("x").YName("yValue").DataSource("ViewBag.dataSource").Name("Germany")
    .Add();
    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    ).PrimaryYAxis(py =>
py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true).Header("Temperature")).Render()
    @Html.EJS().Chart("container1").Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Width(2)

        .XName("x").YName("yValue").DataSource("ViewBag.dataSource").Name("Germany")
        .Add();
        }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

        ).PrimaryYAxis(py =>
py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
        ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true).Header("Temperature")).Render()
    <div>
        @Html.EJS().Button("togglebtn").IsPrimary(true).IconCss("e-icons e-play-
        icon").Content("Export").CssClass("e-flat").Render()
    </div>
    <script>
        document.getElementById('togglebtn').onclick = function () {
            var chart = document.getElementById('container').ej2_instances[0];
            var chart1 = document.getElementById('container1').ej2_instances[0];
            chart.export('PNG', 'chart', 'Landscape', [chart, chart1]);
        };
    </script>

```

MULTI-EXPORT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "USA", yValue= 46 },
        new ColumnChartData { x= "GBR", yValue= 27 },
        new ColumnChartData { x= "CHN", yValue= 26 },
        new ColumnChartData { x= "UK", yValue= 26 },
        new ColumnChartData { x= "AUS", yValue= 26 },
        new ColumnChartData { x= "IND", yValue= 26 },
        new ColumnChartData { x= "DEN", yValue= 26 },
        new ColumnChartData { x= "MEX", yValue= 26 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
}

```

Appearance in MVC Chart component

Chart theme customization

You can customize the default theme of the chart **Theme** property. There are 9 themes available. They are, **Material**, **Fabric**, **Bootstrap**, **HighContrastLight**, **MaterialDark**, **FabricDark**, **HighContrast**, **BootstrapDark** and **Bootstrap4**.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Theme("HighContrast").Render()

```

THEME.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
    }
}

```



```

        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Custom color palette

You can customize the default color of series or points by providing a custom color palette of your choice by using the [Palettes](#) property.

CSHTML

```

@(Html.EJS().Chart("container").Title("Olympic Medal Counts - RIO").PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("country").
    YName("gold").
    DataSource(ViewBag.dataSource).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("country").
    YName("silver").
    DataSource(ViewBag.dataSource).Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("country").
    YName("bronze").
    DataSource(ViewBag.dataSource).Add();
}).
    Palettes(ViewBag.palletes).Render()
)

```

CUSTOM.CS

```

public ActionResult Index()
{
    var palette = new string[] { "#E94649", "#F6B53F", "#6FAAB0" };
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50, silver=70,
bronze=45 },
        new ColumnChartData{ country="China", gold=40, silver= 60,
bronze=55 },
        new ColumnChartData{ country= "Japan", gold=70, silver= 60,
bronze=50 },
        new ColumnChartData{ country= "Australia", gold=60, silver=
56, bronze=40 },
    }
}

```

```

        new ColumnChartData{ country= "France", gold=50, silver= 45,
        bronze=35 },
        new ColumnChartData{ country= "Germany", gold=40, silver=30,
        bronze=22 },
        new ColumnChartData{ country= "Italy", gold=40, silver=35,
        bronze=37 },
        new ColumnChartData{ country= "Sweden", gold=30, silver=25,
        bronze=27 }
    };
    ViewBag.dataSource = chartData;
    ViewBag.palletes = palette;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Data point customization

The color of individual data point or data points within a range can be customized using the options below.

Point color mapping

You can bind the color for the points from [DataSource](#) for the series using [PointColorMapping](#) property.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("y").CornerRadius(cr => cr.TopLeft(10).TopRight(10)).
        DataSource(ViewBag.dataSource).PointColorMapping("color").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg =>
mg.Width(0)).PrimaryYAxis(py => py.LineStyle(mg =>
mg.Width(0)).MajorTickLines(mg => mg.Width(0)).MinorTickLines(mg =>
mg.Width(0)).LabelFormat("{value}°C")).
.Title("USA CLIMATE - WEATHER BY MONTH").ChartArea(area => area.Border(br =>
br.Width(0))).Render()
)
)

```

POINT-COLOR.CS

```

public IActionResult Index()
{
    List<RangeColorMappingData> chartData = new
List<RangeColorMappingData>
    {
        new RangeColorMappingData { x= "Jan", y= 6.96, color=
"red" },

```

```

        new RangeColorMappingData { x= "Feb", y= 8.9, color =
"blue" },
        new RangeColorMappingData { x= "Mar", y= 12, color =
"orange" },
        new RangeColorMappingData { x= "Apr", y= 17.5, color =
"aqua" },
        new RangeColorMappingData { x= "May", y= 22.1, color =
"grey" }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class RangeColorMappingData
{
    public string x;
    public double y;
    public string color;
}

```

Range color mapping

You can differentiate data points based on their y values using [RangeColorSettings](#) in the chart.

CSHTML

```

@(Html.EJS().Chart("container").SelectionMode(Syncfusion.EJ2.Charts.Selectio
nMode.Point).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).XName("x").YName("
y").CornerRadius(cr => cr.TopLeft(10).TopRight(10)).
    DataSource(ViewBag.dataSource).Name("USA").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg =>
mg.Width(0)).PrimaryYAxis(py => py.LineStyle(mg =>
mg.Width(0)).MajorTickLines(mg => mg.Width(0)).MinorTickLines(mg =>
mg.Width(0)).LabelFormat("{value}°C")).LegendSettings(legend =>
legend.Mode(Syncfusion.EJ2.Charts.LegendMode.Range)
    .Visible(true).ToggleVisibility(false)).RangeColorSettings(ranges =>
    {
        ranges.Label("1°C to
10°C").Start(1).End(10).Colors(ViewBag.color1).Add();
        ranges.Label("11°C to
20°C").Start(11).End(20).Colors(ViewBag.color2).Add();
        ranges.Label("21°C to
30°C").Start(21).End(30).Colors(ViewBag.color3).Add();
    }).Title("USA CLIMATE - WEATHER BY MONTH").ChartArea(area =>
area.Border(br => br.Width(0)).Render()
)
)

```

RANGE-COLOR.CS

```

public IActionResult Index()
{
    string[] color1 = { "#F9D422" };
    string[] color2 = { "#F28F3F" };
    string[] color3 = { "#E94F53" };
}

```

```

        List<RangeColorMappingData> chartData = new
List<RangeColorMappingData>
    {
        new RangeColorMappingData { x= "Jan", y= 6.96},
        new RangeColorMappingData { x= "Feb", y= 8.9},
        new RangeColorMappingData { x= "Mar", y= 12},
        new RangeColorMappingData { x= "Apr", y= 17.5},
        new RangeColorMappingData { x= "May", y= 22.1},
        new RangeColorMappingData { x= "June", y= 25},
        new RangeColorMappingData { x= "July", y= 29.4},
        new RangeColorMappingData { x= "Aug", y= 29.6},
        new RangeColorMappingData { x= "Sep", y= 25.8},
        new RangeColorMappingData { x= "Oct", y= 21.1},
        new RangeColorMappingData { x= "Nov", y= 15.5},
        new RangeColorMappingData { x= "Dec", y= 9.9}
    };
    ViewBag.dataSource = chartData;
    ViewBag.color1 = color1;
    ViewBag.color2 = color2;
    ViewBag.color3 = color3;
    return View();
}
public class RangeColorMappingData
{
    public string x;
    public double y;
}

```

Point level customization

Marker, datalabel and fill color of each data point can be customized with [PointRender](#) and [TextRender](#) event.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts - RIO").PointRender("point").Render()
}
<script>
function point(args) {
    args.fill = colors[args.point.index];
}
</script>

```

POINT.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

<!-- markdownlint-disable MD036 -->

Chart area customization

<!-- markdownlint-disable MD036 -->

Customize the chart background

<!-- markdownlint-disable MD013 -->

Using [Background](#) and [Border](#) properties, you can change the background color and border of the chart.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area =>
area.Border(ViewBag.ChartBorder)).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()

```

AREA.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },

```

```

        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country="Japan", gold=70 },
        new ColumnChartData{ country="Australia", gold=60},
        new ColumnChartData{ country="France", gold=50 },
        new ColumnChartData{ country="Germany", gold=40 },
        new ColumnChartData{ country="Italy", gold=40 },
        new ColumnChartData{ country="Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Chart margin

You can set margin for chart from its container through [Margin](#) property.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Width(2).Add();
}).
Margin(ViewBag.margin)
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()

```

MARGIN.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country="USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country="Japan", gold=70 },
        new ColumnChartData{ country="Australia", gold=60},
        new ColumnChartData{ country="France", gold=50 },
        new ColumnChartData{ country="Germany", gold=40 },
        new ColumnChartData{ country="Italy", gold=40 },
        new ColumnChartData{ country="Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{

```

```

        public string country;
        public double gold;
    }

```

Chart area customization

Using [Background](#) and [Border](#) properties, you can change the background color and border of the chart area. Width for the chart area can be customized using [Width](#) property.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
ChartArea(ca => ca.Background('skyblue').Width('90%')).
Title("Olympic Medal Counts - RIO").Render()

```

BACKGROUND.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Animation

You can customize animation for a particular series using [Animation](#) property. You can enable or disable animation of the series using [Enable](#) property, [Duration](#) specifies the duration of an animation and [Delay](#) allows us to start the animation at desire time.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").Render()
```

ANIMATION.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}
```

Fluid animation

Fluid animation used to animate series with updated dataSource continues animation rather than animation whole series. You can customize animation for a particular series using `[Animate]` method.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).

    .CornerRadius(cr=>cr.BottomLeft(10).BottomRight(10).TopLeft(10).TopRight(10)
)
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Width(2).Add();
```



```

    }).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Render()
<script>
    var count = 0;
    var loaded = function (args) {
        args.chart.loaded = null;
        setInterval(
            function () {
                if (count === 0) {

                    args.chart.series[0].dataSource = [
                        { x: 'Tea', y: 206,
text: 'Bangaladesh' },
                        { x: 'Misc', y: 123,
text: 'Bhutn' },
                        { x: 'Fish', y: 48,
text: 'Nepal' },
                        { x: 'Egg', y: 240,
text: 'Thiland' },
                        { x: 'Fruits', y: 170,
text: 'Malaysia' }

                    ];
                    args.chart.animate();
                    count++;
                }
                else if (count === 1) {

                    args.chart.series[0].dataSource = [
                        { x: 'Tea', y: 86,
text: 'Bangaladesh' },
                        { x: 'Misc', y: 173,
text: 'Bhutn' },
                        { x: 'Fish', y: 188,
text: 'Nepal' },
                        { x: 'Egg', y: 109,
text: 'Thiland' },
                        { x: 'Fruits', y: 100,
text: 'Malaysia' }

                    ];
                    args.chart.animate();
                    count++;
                }
                else if (count === 2) {

                    args.chart.series[0].dataSource = [
                        { x: 'Tea', y: 156,
text: 'Bangaladesh' },
                        { x: 'Misc', y: 33,
text: 'Bhutn' },
                        { x: 'Fish', y: 260,
text: 'Nepal' },
                        { x: 'Egg', y: 200,
text: 'Thiland' },
                        { x: 'Fruits', y: 30,
text: 'Malaysia' }

                    ];

```

```

        args.chart.animate();
        count = 0;
    }
    }, 2000);
}
</script>
}

```

FLUID.CS

```

public IActionResult Index()
{
    List<RoundedColumnChartData> chartData = new
List<RoundedColumnChartData>
{
    new RoundedColumnChartData { x= "BGD", y= 106, text=
"Bangaladesh" },
    new RoundedColumnChartData { x= "BTN", y= 103, text=
"Bhutn" },
    new RoundedColumnChartData { x= "NPL", y= 198, text=
"Nepal" },
    new RoundedColumnChartData { x= "THA", y= 189, text=
"Thiland" },
    new RoundedColumnChartData { x= "MYS", y= 250, text=
"Malaysia" }
};
    ViewBag.dataSource = chartData;
    ViewBag.font = new { fontWeight = "600", color = "#ffffff" };
    return View();
}

public class RoundedColumnChartData
{
    public string x;
    public double y;
    public string text;
}

```

Chart title

Chart can be given a title using [Title](#) property, to show the information about the data plotted.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
Marker(ViewBag.Marker).
XName("x").
YName("yValue").
DataSource(ViewBag.dataSource).
Name("Gold").
Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO").TitleStyle(ViewBag.style).Render()

```

TITLE.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

Title position

By using the [Position](#) property in [TitleStyle](#), you can position the [Title](#) at left, right, top or bottom of the chart. The title is positioned at the top of the chart, by default.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("country").
    YName("gold").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts -
RIO").TitleStyle(tl=>tl.Position(Syncfusion.EJ2.Charts.TitlePosition.Bottom)
).Render()

```

TITLEPOSITION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },

```

```

        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
    public double gold;
}

```

The custom option helps you to position the title anywhere in the chart using [X](#) and [Y](#) coordinates.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("country").
    YName("gold").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts -
RIO").TitleStyle(tl=>tl.Position(Syncfusion.EJ2.Charts.TitlePosition.Custom)
.X(300).Y(60)).Render()

```

TITLEOPTION.CS

```

public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ColumnChartData
{
    public string country;
}

```

```
        public double gold;
    }
```

Title alignment

You can align the title to the near, far, or center of the chart using the [TextAlignment](#) property.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    XName("country").
    YName("gold").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
    Title("Olympic Medal Counts -
RIO").TitleStyle(tl=>tl.Position(Syncfusion.EJ2.Charts.TitlePosition.Bottom)
.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Far)).Render()
```

TITLEALIGNMENT.CS

```
public IActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}
```

Chart subTitle

Chart can be given a subtitle using [SubTitle](#) property, to show the information about the data plotted.

CSHTML

```
@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    SubTitle("Olympic Medal Counts - RIO").
```

```

XName("country").
YName("gold").
DataSource(ViewBag.dataSource).
Name("Gold").
Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
Title("Olympic Medal Counts - RIO")
.SubTitle("(1975-2010)")
.Render()
)

```

SUBTITLE.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData{ country= "USA", gold=50 },
        new ColumnChartData{ country="China", gold=40 },
        new ColumnChartData{ country= "Japan", gold=70 },
        new ColumnChartData{ country= "Australia", gold=60},
        new ColumnChartData{ country= "France", gold=50 },
        new ColumnChartData{ country= "Germany", gold=40 },
        new ColumnChartData{ country= "Italy", gold=40 },
        new ColumnChartData{ country= "Sweden", gold=30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string country;
    public double gold;
}

```

<!-- markdownlint-disable MD036 -->

Rendering Types

Chart uses following two rendering methods.

- SVG
- Canvas

SVG

SVG is used to render Chart by default for all browsers except IE8 and old versions.

Canvas

You can switch between SVG and Canvas rendering by using the `EnableCanvas` option. The canvas mode rendering is used in the following scenarios,

- Plotting large number of data points.

- Performing high frequency live updates.

Limitations

- Animation is not supported.

Accessibility in ASP.NET MVC Chart component

The Chart component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Chart component is outlined below.

Accessibility Criteria	Compatibility
--	--
WCAG 2.2 Support	
Section 508 Support	
Screen Reader Support	
Right-To-Left Support	
Color Contrast	
Mobile Device Support	
Keyboard Navigation Support	
Accessibility Checker Validation	
Axe-core Accessibility Validation	

```

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

```

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Chart component:

- img (role)
- button (role)
- region (role)
- aria-label (attribute)
- aria-hidden (attribute)
- aria-pressed (attribute)

Keyboard interaction

The Chart component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Chart component.

| **Press** | **To do this** |

| --- | --- |

| Alt + J | Moves the focus to the chart element. |

| Tab | Moves the focus to the next element in the chart. |

| Shift + Tab | Moves the focus to the previous element in the chart. |

| Down Arrow | Moves the focus to the data point left side from the selected point. |

| Up Arrow | Moves the focus to the data point right side from the selected point. |

| Left Arrow | Moves the focus to the next series in the chart. |

| Right Arrow | Moves the focus to the previous series in the chart. |

| ESC | Cancel the tooltip for the data point. |

| Enter/Space | Selects the data point in the series. |

| Down/Left Arrow | Moves the focus to the legend left side from the selected legend. |

| Up/Right Arrow | Moves the focus to the legend right side from the selected legend. |

| Enter/Space | Toggles the visibility of the corresponding series. |

| Ctrl + + | Zoom in the chart. |

| Ctrl + - | Zoom out the chart. |

| Down/Up Arrow | Pan the chart vertically. |

| Left/Right Arrow | Pan the chart horizontally. |

| R | Reset the zoomed chart. |

| Ctrl + P | Prints the Chart. |

Ensuring accessibility

The Chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Chart component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Internationalization

Chart provide supports for internationalization for below chart elements.

- Datalabel.
- Axis label.
- Tooltip.

For more information about number and date formatter you can refer [internationalization](#).

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing an component that works in different cultures/locales. Internationalization library is used to globalize number, date, time values in Chart component using `LabelFormat` property in axis.

Numeric Format

In the below example axis, point and tooltip labels are globalized to EUR.

CSHTML

```
@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).
    Marker(ViewBag.Marker).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Width(2).Add();
}).
PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).
PrimaryYAxis(py => py.LabelFormat("n1")).
Title("Olympic Medal Counts - RIO").Render()
```

NUMBER-FORMAT.CS

```

public ActionResult Index()
{
    List<InternalChartData> chartData = new List<InternalChartData>
    {
        new InternalChartData {x= 1900, y= 4, y1= 2.6 },
        new InternalChartData{ x= 1920, y= 3.0, y1= 2.8 },
        new InternalChartData{ x= 1940, y= 3.8, y1= 2.6},
        new InternalChartData{ x= 1960, y= 3.4, y1= 3 },
        new InternalChartData{ x= 1980, y= 3.2, y1= 3.6 },
        new InternalChartData{ x= 2000, y= 3.9, y1= 3 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class InternalChartData
{
    public double x;
    public double y;
    public double y1;
}

```

Localization

Localization library allows to localize the default text content of Chart. In Chart component, it has the static text on some features(like zooming toolbars) and this can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

Locale key words	Text to display
Zoom	Zoom
ZoomIn	ZoomIn
ZoomOut	ZoomOut
Reset	Reset
Pan	Pan
ResetZoom	Reset Zoom

To load translation object in an application use load function of L10n class.

For more information about localization, refer this [localization](#)

CSHTML

```

@Html.EJS().Chart("container").Locale("fr-FR").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
    .Width(2)
    .XName("x").YName("y")
    .DataSource("series1")
    .Name("Germany").Add();
}

```

```

    }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .Skeleton("yMMM")
    ).PrimaryYAxis(py =>
py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    ).Title("Inflation - Consumer Price")
    .ZoomSettings(z => z.EnableMouseWheelZooming(true).
    EnablePinchZooming(true).EnableSelectionZooming(true).
    Mode(Syncfusion.EJ2.Charts.ZoomMode.X)
    ).Render()
<script>
var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 100; i++) {
    if (Math.random() > 0.5) {
        value += Math.random();
    }
    else {
        value -= Math.random();
    }
    point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
    series1.push(point1);
}
ej.base.L10n.load({
    "fr-FR": {
        "chart": {
            "ZoomIn": "Zoom +",
            "ZoomOut": "Zoom -",
            "Pan": "Se déplacer",
            "Reset": "Réinitialiser"
        }
    }
});
ej.base.setCulture("fr-FR");
</script>

```

LOCALIZATION.CS

```

public ActionResult Index()
{
    List<InternalChartData> chartData = new List<InternalChartData>
    {
        new InternalChartData{ x= 1900, y= 4, y1= 2.6 },
        new InternalChartData{ x= 1920, y= 3.0, y1= 2.8 },
        new InternalChartData{ x= 1940, y= 3.8, y1= 2.6},
        new InternalChartData{ x= 1960, y= 3.4, y1= 3 },
        new InternalChartData{ x= 1980, y= 3.2, y1= 3.6 },
        new InternalChartData{ x= 2000, y= 3.9, y1= 3 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class InternalChartData
{

```

```

public double x;
public double y;
public double y1;
}

```

Migration from Essential JS 1

This article describes the API migration process of Chart component from Essential JS 1 to Essential JS 2.

Annotations

Behavior	API in Essential JS 1	API in Essential JS 2
rotation of annotation	Property:Annotations.Angle@ (Html.EJ().Chart("chartContainer").Annotations(an => { .Angle(270).Add(); }))	Not applicable
annotations	Property:Annotations.Content@ (Html.EJ().Chart("chartContainer").Annotations(an => { an.Content("watermark").Add(); }))	Property:Annotations.Content@Html.EJS().Chart("container").Annotations(an => { an.Content(ViewBag.content).Add(); })
coordinate unit for annotation	Property:Annotations.CoordinateUnit@ (Html.EJ().Chart("chartContainer").Annotations(an => { an.CoordinateUnit(CoordinateUnit.Pixels).Add(); }))	Property:Annotations.CoordinateUnits@Html.EJS().Chart("container").Annotations(an => { an.CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Add(); })
horizontal alignment for annotation	Property:Annotations.HorizontalAlignment@ (Html.EJ().Chart("chartContainer").Annotations(an => { an.HorizontalAlignment(HorizontalAlignment.Left).Add(); }))	Property:Annotations.HorizontalAlignment@Html.EJS().Chart("container").Annotations(an => { an.HorizontalAlignment(Syncfusion.EJ2.Charts.Position.Near).Add(); })
margin for annotation	Property:Annotations.Margin@ (Html.EJ().Chart("chartContainer").Annotations(an => { an.Margin(mr=>mr.Right(40)).Add(); }))	Not applicable
Opacity for annotation	Property:Annotations.Opacity@ (Html.EJ().Chart("chartContainer").Annotations(an => { an.Opacity(0.2).Add(); }))	Not applicable
Region for annotation with respect	Property:Annotations.Region@ (Html.EJ().Chart("chartContainer").Annotations(an => { an.Region(Region.Series).Add(); }))	Property:Annotations.Region@Html.EJS().Chart("container").Annotations(an => { an.Region(Region.Series).Add(); })

to chart or series		
verticalAlignment for annotation	Property:Annotations.VerticalAlignment@(Html.EJ().Chart("chartContainer").Annotations(an => { an.VerticalAlignment(VerticalAlignment.Middle).Add();}))	Property:Annotations.VerticalAlignment@Html.EJS().Chart("container").Annotations(an => { an.VerticalAlignment(Syncfusion.EJ2.Charts.Position.Top).Add();})
Visibility of annotations	Property:Annotations.Visible@(Html.EJ().Chart("chartContainer").Annotations(an => { an.Visible(true).Add();}))	Not applicable
X offset for annotation	Property:Annotations.X@(Html.EJ().Chart("chartContainer").Annotations(an => { an.X(170).Add();}))	Property:Annotations.X@Html.EJS().Chart("container").Annotations(an => { an.X("170").Add();})
X axis name in which annotation to be rendered	Property:Annotations.XAxisName@(Html.EJ().Chart("chartContainer").Annotations(an => { an.XAxisName("xAxis").Add();}))	Property:Annotations.XAxisName@Html.EJS().Chart("container").Annotations(an => { an.XAxisName("xAxis").Add();})
Y offset for annotation	Property:Annotations.Y@(Html.EJ().Chart("chartContainer").Annotations(an => { an.Y("50").Add();}))	Property:Annotations.Y@Html.EJS().Chart("container").Annotations(an => { an.Y("50").Add();})
Y axis name in which annotation to be rendered	Property:Annotations.YAxisName@(Html.EJ().Chart("chartContainer").Annotations(an => { an.YAxisName("yAxis").Add();}))	Property:Annotations.YAxisName@Html.EJS().Chart("container").Annotations(an => { an.YAxisName("yAxis").Add();})

Columns

Behaviour	API in Essential JS 1	API in Essential JS 2
Columns in chart	Property:ColumnDefinitions@(Html.EJ().Chart("chartContainer").ColumnDefinitions())	Property:Columns@Html.EJS().Chart("container").Columns()

unit	Property:ColumnDefinitions.Unit@(Html.EJ().Chart("chartContainer").ColumnDefinitions(cdn => { cdn.Unit("percentage").Add(); })	Not applicable
width of columns in chart	Property:ColumnDefinitions.ColumnWidth@(Html.EJ().Chart("chartContainer").ColumnDefinitions(cdn => { cdn.ColumnWidth(50).Add(); })	Property:Columns.Width@Html.EJS().Chart("container").Columns(columns => { columns.width("50%").Add(); })
Line customization	Property:LineColor, LineWidth@(Html.EJ().Chart("chartContainer").ColumnDefinitions(cdn => { cdn.ColumnWidth(50).LineColor('red').LineWidth(2).Add(); })	Property:Columns.Border@Html.EJS().Chart("container").Columns(columns => { columns.Width("50%").Border(br => br.Width("2").Color("red")).Add(); })

CommonSeriesOptions

Behaviour	API in Essential JS 1	API in Essential JS 2
commonSeriesOptions	Property:CommonSeriesOptions@(Html.EJ().Chart("chartContainer").CommonSeriesOptions())	Not applicable

Crosshair

Behaviour	API in Essential JS 1	API in Essential JS 2
crosshair	Property:Crosshair.Visible@(Html.EJ().Chart("chartContainer").Crosshair(crosshair=>crosshair.Visible(true)))	Property:Crosshair.Enable@Html.EJS().Chart("container").Crosshair(cr=>cr.Enable(true))
trackball TooltipSettings	Property:TrackballTooltipSettings@(Html.EJ().Chart("chartContainer").Crosshair(crosshair => crosshair.TrackballTooltipSettings(tr => tr.Border(br => br.Width(2)))	Not applicable
marker	Property:Marker@(Html.EJ().Chart("chartContainer").Crosshair(crosshair => crosshair.Marker(mr => mr.Border(br => br.Width(2)))	Not applicable
crosshair line style	Property:Line@(Html.EJ().Chart("chartContainer").Crosshair(crosshair => crosshair.Line(line=>line.Color("grey").Width(2)))	Property:Line@Html.EJS().Chart("container").Crosshair(crosshair => crosshair.Line(line => line.Border(br => br.Width(2).Color("grey")))

type	<code>Property:Type@(Html.EJ().Chart("chartContainer").Crosshair(crosshair => crosshair.Type("trackball")))</code>	Not applicable
------	---	----------------

3D chart

Behaviour	API in Essential JS 1	API in Essential JS 2
3d chart	<code>Property:Enable3D@(Html.EJ().Chart("chartContainer").Enable3D(true))</code>	Not applicable
Rotation of 3d chart	<code>Property:EnableRotation@(Html.EJ().Chart("chartContainer").EnableRotation(true))</code>	Not applicable
depth	<code>Property:Depth@(Html.EJ().Chart("chartContainer").Depth(120))</code>	Not applicable

Canvas rendering

Behaviour	API in Essential JS 1	API in Essential JS 2
canvas rendering	<code>Property:EnableCanvasRendering@(Html.EJ().Chart("chartContainer").EnableCanvasRendering(true))</code>	Not applicable

Indicators

Behaviour	API in Essential JS 1	API in Essential JS 2
Type of Indicator	<code>Property:Type@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .Type(ChartIndicatorType.AccumulationDistribution).Add(); }))</code>	<code>Property:Type@Html.EJS().Chart("container").Indicators(id =>{id.Type(Syncfusion.EJ2.Charts.TechnicalIndicators.AccumulationDistribution).Add();})</code>
Period for indicator	<code>Property:Period@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .Period(3).Add(); }))</code>	<code>Property:Period@Html.EJS().Chart("container").Indicators(id =>{id.Period(3).Add();})</code>
%K value in stochasti	<code>Property:KPeriod@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .KPeriod(14).Add(); }))</code>	<code>Property:KPeriod@Html.EJS().Chart("container").Indicators(id =>{id.KPeriod("14").Add();})</code>

c indicator		
%D value in stochasti c indicator	Property:DPeriod@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .DPeriod("3").Add(); })))	Property:DPeriod@Html.EJS().Chart("container").Indicators(id =>{id.DPeriod("3").Add();})
Shows overSold /overBou ght values	Not Applicable	Property:ShowZones@Html.EJS().Chart("container").Indicators(id =>{id.ShowZones(true).Add();})
Overbou ght value for RSI and stochasti c indicator	Not Applicable	Property:OverBought@Html.EJS().Chart("container").Indicators(id =>{id.OverBought("80").Add();})
Oversold value for RSI and stochasti c indicator	Not Applicable	Property:OverSold@Html.EJS().Chart("container").Indicators(id =>{id.OverSold("20").Add();})
Standard deviation for Bollinger bands	Property:StandardDeviations@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .StandardDeviations("2").Add(); })))	Property:StandardDeviation@Html.EJS().Chart("container").Indicators(id =>{id.StandardDeviation("2").Add();})
Field for indicator	Property:Field@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .Field("Close").Add(); })))	Property:Field@Html.EJS().Chart("container").Indicators(id =>{id.Field(Syncfusion.EJ2.Charts.FinancialDataFields.Close).Add();})
Slow period for MACD indicator	Property:ShortPeriod@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .ShortPeriod("12").Add(); })))	Property:slowPeriod@Html.EJS().Chart("container").Indicators(id =>{id.slowPeriod("12").Add();})
Fast period for	Property:LongPeriod@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind .LongPeriod("26").Add(); })))	Property:FastPeriod@Html.EJS().Chart("container").Indicators(id =>{id.FastPeriod("26").Add();})

MACD indicator		
Line style for MACD indicator	Property:MacdLine@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind.MacdLine(ma => ma.Width("2").Fill("red")).Add(); })))	Property:MacdLine@Html.EJS().Chart("container").Indicators(id =>{id.MacdLine(ma => ma.Width("2").Fill("red")).Add();})
Type of MACD indicator	Property:Type@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind.Type(ChartIndicatorType.MACD).Add(); })))	Property:MacdType@Html.EJS().Chart("container").Indicators(id =>{id.MacdType(Syncfusion.EJ2.Charts.MacdType.Both).Add();})
Color of the positive bars in Macd indicators	Not Applicable	Property:MacdPositiveColor@Html.EJS().Chart("container").Indicators(id =>{id.MacdPositiveColor("red").Add();})
Color of the negative bars in Macd indicators	Not Applicable	Property:MacdNegativeColor@Html.EJS().Chart("container").Indicators(id =>{id.MacdNegativeColor("red").Add();})
Color for Bollinger bands	Not Applicable	Property:BandColor@Html.EJS().Chart("container").Indicators(id =>{id.BandColor("red").Add();})
Appearance of upper line in indicator	Property:UpperLine@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind.UpperLine(up => up.Fill("red").Width("2")).Add(); })))	Property:UpperLine@Html.EJS().Chart("container").Indicators(id =>{id.UpperLine(up => up.Fill("red").Width("2")).Add();})
Appearance of lower line in indicator	Property:LowerLine@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind.LowerLine(lo => lo.Fill("red").Width("2")).Add(); })))	Property:LowerLine@Html.EJS().Chart("container").Indicators(id =>{id.LowerLine(lo => lo.Fill("red").Width("2")).Add();})
Appearance of period	Property:PeriodLine@(Html.EJ().Chart("chartContainer").Indicators(ind => { ind.PeriodLine(pl => pl.Fill("red").Width("2")).Add(); })))	Property:PeriodLine@Html.EJS().Chart("container").Indicators(id =>{id.PeriodLine(pl => pl.Fill("red").Width("2")).Add();})

line in indicator		
Name of the series for which indicator has to be drawn	Property:SeriesName@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.SeriesName("series1").Add(); }));	Property:SeriesName@Html.EJS().Chart("container").Indicators(id =>{id.SeriesName("series1").Add();})
Options to customize the histogram in MACD indicator	Property:Histogram@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.Histogram().Add(); }));	Not Applicable
Enabling animation	Property:EnableAnimation@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.enableAnimation(true).Add(); }));	Property:Animation.Enable@Html.EJS().Chart("container").Indicators(id =>{id.Animation(an => an.Enable(true)).Add();})
Animation duration	Property:AnimationDuration@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.animationDuration("3000").Add(); }));	Property:Animation.Duration@Html.EJS().Chart("container").Indicators(id =>{id.Animation(an => an.Duration("3000")).Add();})
Tooltip	Property:Tooltip.Visible@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.Tooltip(tp => tp.Visible(true)).Add(); }));	Not Applicable
Trigger value of MACD indicator	Property:Trigger@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.Trigger("14").Add(); }));	Not Applicable
Fill color for indicator	Property:Fill@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.Fill("red").Add(); }));	Property:Fill@Html.EJS().Chart("container").Indicators(id =>{id.Fill("red").Add();})
Width for indicator	Property:Width@ (Html.EJ() .Chart("chartContainer").Indicators(ind => { ind.Width("2").Add(); }));	Property:Width@Html.EJS().Chart("container").Indicators(id =>{id.Width("2").Add();})

xAxis Name of indicator	Property:XAxisName@ (Html.EJ().Chart("chartContainer").Indicators(ind => { ind.XAxisName("").Add(); })))	Property:XAxisName@Html.EJS().Chart("container").Indicators(id =>{id.XAxisName("").Add();})
yAxis Name of indicator	Property:YAxisName@ (Html.EJ().Chart("chartContainer").Indicators(ind => { ind.YAxisName("").Add(); })))	Property:YAxisName@Html.EJS().Chart("container").Indicators(id =>{id.YAxisName("").Add();})
Visibility of indicator	Property:Visibility@ (Html.EJ().Chart("chartContainer").Indicators(ind => { ind.Visibility(true).Add(); })))	Not Applicable

Legend

Behavior	API in Essential JS 1	API in Essential JS 2
Default legend	Property:Legend.Visible@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Visible(true)))	Property:LegendSettings.Visible@Html.EJS().Chart("container").LegendSettings(legend => legend.Visible(true))
Legend height	Property:Size.Height@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Size(sz => sz.Height("50"))))	Property:Height@Html.EJS().Chart("container").LegendSettings(legend => legend.Height("50"))
Legend width	Property:Size.Width@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Size(sz => sz.Width("50"))))	Property:Width@Html.EJS().Chart("container").LegendSettings(legend => legend.Width("50"))
Legend location in chart	Property:Location@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Location(loc => loc.X("3").Y("45"))))	Property:Location@Html.EJS().Chart("container").LegendSettings(le => le.Location(loc => loc.X("3").Y("45")))
Legend position in chart	Property:Position@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Position(LegendPosition.Top)))	Property:Position@Html.EJS().Chart("container").LegendSettings(le => le.Position(Syncfusion.EJ2.Charts.LegendPosition.Top))

Legend padding	Not applicable	Property:Padding@Html.EJS().Chart("container").LegendSettings(le =>le.Padding("8"))
Legend alignment	Property:Alignment@(Html.EJ().Chart("chartContainer").Legend(lg=>lg.Alignment(Syncfusion.JavaScript.DataVisualization.TextAlignment.Far)))	Property:Alignment@Html.EJS().Chart("container").LegendSettings(le =>le.Alignment(Syncfusion.EJ2.Charts.Alignment.Far))
Text style for legend	Property:Font@(Html.EJ().Chart("chartContainer").Legend(lg=>lg.Font(font=>font.FontFamily("Segoe UI").FontStyle(CharFontStyle.Italic).FontWeight(CharFontWeight.Bold).Size("12px"))))	Property:Font@Html.EJS().Chart("container").LegendSettings(le =>le.Font(ViewBag.font))
Shape height of legend	Property:ItemStyle.Height@(Html.EJ().Chart("chartContainer").Legend(lg=>lg.ItemStyle(item=>item.Height(13))))	Property:ShapeHeight@Html.EJS().Chart("container").LegendSettings(le =>le.ShapeHeight(20))
Shape width of legend	Property:ItemStyle.Width@(Html.EJ().Chart("chartContainer").Legend(lg=>lg.ItemStyle(item=>item.Width("20"))))	Property:ShapeWidth@Html.EJS().Chart("container").LegendSettings(le =>le.ShapeWidth("20"))
Shape border of legend	Property:ItemStyle.Border@(Html.EJ().Chart("chartContainer").Legend(lg=>lg.ItemStyle(item=>item.Border(br=>br.Width(1).Color("#FF0000"))))	Not Applicable
Shape padding of	Property:ItemPadding@(Html.EJ().Chart("chartContainer").Legend(lg=>lg.ItemPadding(15)))	Property:ShapePadding@Html.EJS().Chart("container").LegendSettings(le =>le.ShapePadding("20"))

Legend		
Background of legend	Property:Background@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Background("transparent"))	Property:Backgorund@Html.EJS() .Chart("container").LegendSettings(le =>le.Backgorund("transparent"))
Opacity of legend	Property:Opacity@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Opacity("0.3")))	Property:Opacity@Html.EJS().Chart("container").LegendSettings (le =>le.Opacity("0.3"))
Toggle visibility of series while legend click	Property:ToggleSeriesVisibility@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.ToggleSeriesVisibility(true)))	Property:ToggleVisibility@Html.EJS().Chart("container").LegendSettings (le =>le.ToggleVisibility(true))
Title for legend	Property:Title@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.Title()))	Not applicable
Text Overflow for legend	Property:TextOverflow@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.TextOverflow(TextOverflow.Trim)))	Property:TextStyle.TextOverFlow@ Html.EJS().Chart("container").LegendSettings(le =>le.TextStyle(tx =>tx.TextOverFlow("Trim")))
Text width for	Property:TextWidth@ (Html.EJ().Chart("chartContainer").Legend(lg=>lg.TextWidth("3")))	Not applicable

legend while setting text overflow		
Scrollbar for legend	Property:EnableScrollBar@ (Html.EJ() .Chart ("chartContainer") .Legend(lg=>lg.EnableScrollBar(true)))	Not applicable
Row count for legend	Property:RowCount@ (Html.EJ() .Chart ("chartContainer") .Legend(lg=>lg.RowCount("2")))	Not applicable
Column count for legend	Property:ColumnCount@ (Html.EJ() .Chart ("chartContainer") .Legend(lg=>lg.ColumnCount("2")))	Not applicable
Color for legend items	Property:Fill@ (Html.EJ() .Chart ("chartContainer") .Legend(lg=>lg.Fill("red")))	Not applicable

primaryXAxis

Behaviour	API in Essential JS 1	API in Essential JS 2
-----------	-----------------------	-----------------------

Alternate grid band	Property:AlternateGridBand@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.AlternateGridBand(alt => alt.even(e => e.Fill("red")))))	Not applicable
Axis line cross value	Property:CrossesAt@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.CrossesAt(0)))	Property:CrossesAt@Html.EJS().Chart("container").PrimaryXAxis(px=>px.CrossesAt(0))
Axis name with which the axis line has to be crossed	Property:CrossesInAxis@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.CrossesInAxis("")))	Property:CrossesInAxis@Html.EJS().Chart("container").PrimaryXAxis(px=>px.CrossesInAxis(""))
Axis elements placed with axis line	Property:ShowNextToAxisLine@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ShowNextToAxisLine(true)))	Property:PlaceNextToAxisLine@Html.EJS().Chart("container").PrimaryXAxis(px=>px.PlaceNextToAxisLine(true))
Axis line style	Property:AxisLine.Color@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.AxisLine(al => al.Color("red"))))	Property:LineStyle.Color@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LineStyle(ls => ls.Color("black")))
Axis line dash array	Property:AxisLine.DashArray@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.AxisLine(al => al.DashArray("10,5"))))	Property:LineStyle.DashArray@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LineStyle(ls => ls.DashArray("10,5")))
Offset for axis	Property:AxisLine.Offset@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.AxisLine(al => al.Offset("10"))))	Property:PlotOffset@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LineStyle(ls => ls.PlotOffset("10")))
Visible of an axis	Property:AxisLine.Visible@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.AxisLine(al => al.Visible(false)))	Property:Visible@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LineStyle(ls => ls.Visible(false))

Width of an axis	<code>Property:AxisLine.Width@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.AxisLine(al => al.Width("2"))))</code>	<code>Property:LineStyle.Width@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LineStyle(ls => ls.Width("2")))</code>
Column index of an axis	<code>Property:ColumnIndex@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ColumnIndex("2")))</code>	<code>Property:ColumnIndex@Html.EJS().Chart("container").PrimaryXAxis(px=>px.ColumnIndex("2"))</code>
Span of an axis to place horizontally or vertically	<code>Property:ColumnSpan@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ColumnSpan("2")))</code>	<code>Property:Span@Html.EJS().Chart("container").PrimaryXAxis(px=>px.Span("2"))</code>
Cross hair label of an axis	<code>Property:CrossHairLabel.Visible(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.CrossHairLabel(cr => cr.Visible(true))))</code>	<code>Property:CrossHairTooltip.Enable@Html.EJS().Chart("container").PrimaryXAxis(px=>px.CrossHairLabel(cr => cr.Enable(true)))</code>
Cross hair label color of an axis	Not applicable	<code>Property:CrossHairTooltip.Fill@Html.EJS().Chart("container").PrimaryXAxis(px=>px.CrossHairLabel(cr => cr.Fill("red")))</code>
Cross hair label text style	Not applicable	<code>Property:CrossHairTooltip.TextStyle@Html.EJS().Chart("container").PrimaryXAxis(px=>px.CrossHairLabel(cr => cr.TextStyle()))</code>
Desired interval count for primary XAxis	<code>Property:DesiredIntervals@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.DesiredIntervals("4")))</code>	<code>Property:DesiredIntervals@Html.EJS().Chart("container").PrimaryXAxis(px=>px.DesiredIntervals("4"))</code>

Edges primary XAxis	Property: EdgeLabelPlacement@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.EdgeLabelPlacement("none")))	Property: EdgeLabelPlacement@Html.EJS().Chart("container").PrimaryXAxis(px=>px.EdgeLabelPlacement("none"))
Enables trim for axis labels	Property: EnableTrim@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.EnableTrim(true)))	Property: EnableTrim@Html.EJS().Chart("container").PrimaryXAxis(px=>px.EnableTrim(true))
Specifies the interval of the axis according to the zoomed data of the chart	Property: EnableAutoIntervalOnZooming@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.EnableAutoIntervalOnZooming(true)))	Property: EnableAutoIntervalOnZooming@Html.EJS().Chart("container").PrimaryXAxis(px=>px.EnableAutoIntervalOnZooming(true))
Title style for primary XAxis	Property: Font@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Font()))	Property: TitleStyle@Html.EJS().Chart("container").PrimaryXAxis(px=>px.TitleStyle())
Indexed for category axis	Property: IsIndexed@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.IsIndexed(true)))	Property: IsIndexed@Html.EJS().Chart("container").PrimaryXAxis(px=>px.IsIndexed(true))
Interval type for date time axis	Property: IntervalType@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.IntervalType("Auto")))	Property: IntervalType@Html.EJS().Chart("container").PrimaryXAxis(px=>px.IntervalType("Auto"))

Inversed axis	Property:IsInversed@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.IsInversed(true)))	Property:IsInversed@Html.EJS().Chart("container").PrimaryXAxis(px=>px.IsInversed(true))
Custom label format	Property:LabelFormat@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LabelFormat('{value}K')))	Property:LabelFormat@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LabelFormat('{value}K'))
labelIntersectAction	Property:LabelIntersectAction@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LabelIntersectAction("trim")))	Property:LabelIntersectAction@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LabelIntersectAction("Trim"))
labelPosition	Property:LabelPosition@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LabelPosition("inside")))	Property:LabelPosition@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LabelPosition("Inside"))
labelPlacement for category axis	Property:LabelPlacement@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LabelPlacement("onTicks")))	Property:LabelPlacement@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LabelPlacement("onTicks"))
Axis label alignment	Property:Alignment@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Alignment("center")))	Not Applicable
Rotation of axis labels	Property:LabelRotation@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LabelRotation("45")))	Property:LabelRotation@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LabelRotation("45"))
Log base value for logarithmic axis	Property:LogBase@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LogBase("10")))	Property:LogBase@Html.EJS().Chart("container").PrimaryXAxis(px=>px.LogBase("10"))
Major grid line visible	Property:MajorGridLines.visible@ (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Visible(true))))	Not Applicable

Width of Major Grid Lines	Property:MajorGridLines.Width@Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Width("2"))))	Property:MajorGridLines.Width@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Width("2"))))
Color of Major Grid Lines	Property:MajorGridLines.Color(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Color("black"))))	Property:MajorGridLines.Color@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Color("black"))))
Dash Array of Major Grid Lines	Property:MajorGridLines.DashArray(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.DashArray("10,2"))))	Property:MajorGridLines.DashArray@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.DashArray("10,2"))))
Opacity of major grid line	Property:MajorGridLines.Opacity(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Opacity("0.5"))))	Not Applicable
Major Tick line	Property:MajorTickLines.Visible(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorTickLines(mtl => mtl.Visible(true))))	Not Applicable
Width of Major Tick Lines	Property:MajorTickLines.Width(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorTickLines(mtl => mtl.Width("2"))))	Property:MajorTickLines.Width@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Width("2"))))
Height of Major Tick Lines	Property:MajorTickLines.Size(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorTickLines(mtl => mtl.Size("2"))))	Property:MajorTickLines.Height@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Height("2"))))
Color of Major Tick Lines	Property:MajorTickLines.Color(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MajorTickLines(mtl => mtl.Color("black"))))	Property:MajorTickLines.Color@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MajorGridLines(mgl => mgl.Color("black"))))
Opacity of major	Property:MajorTickLines.Opacity(Html.EJ().Chart("chartContainer").PrimaryXAxis(p	Not Applicable

Tick line	<code>x=>px.MajorTickLines(mtl => mtl.Opacity("0.5"))))</code>	
maximum labels of primary XAxis	<code>Property:MaximumLabels(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MaximumLabels("5")))</code>	<code>Property:MaximumLabels@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MaximumLabels("5"))</code>
maximum labels width of primary XAxis is to trim	<code>Property:MaximumLabelWidth(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MaximumLabelWidth("40")))</code>	<code>Property:MaximumLabelWidth@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MaximumLabelWidth("40"))</code>
minor grid line	<code>Property:MinorGridLines.Visible(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.Visible(true))))</code>	Not Applicable
Width of minor GridLines	<code>Property:MinorGridLines.Width(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.Width("2"))))</code>	<code>Property:MinorGridLines.Width@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.Width("2")))</code>
Color of minor GridLines	<code>Property:MinorGridLines.Color(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.Color("black"))))</code>	<code>Property:MinorGridLines.Color@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.Color("black")))</code>
Dash Array of minor GridLines	<code>Property:MinorGridLines.DashArray(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.DashArray("10,5"))))</code>	<code>Property:MinorGridLines.DashArray@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.DashArray("10,5")))</code>
Opacity of minor grid line	<code>Property:MinorGridLines.Opacity(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorGridLines(mi => mi.Opacity("0.5"))))</code>	Not Applicable

minor Tick line	Property:MinorTickLines.Visible (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorTickLines(mit => mit.Visible(true))))	Not Applicable
Width of minor TickLines	Property:MinorTickLines.Width (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorTickLines(mit => mit.Width("2"))))	Property:MinorTickLines.Width@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorTickLines(mi => mi.Width("2")))
Height of minor TickLines	Property:MinorTickLines.Size (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorTickLines(mit => mit.Size("2"))))	Property:MinorTickLines.Height@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorTickLines(mi => mi.Height("2")))
Color of minor TickLines	Property:MinorTickLines.Color(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorTickLines(mit => mit.Color("black"))))	Property:MinorTickLines.Color@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorTickLines(mi => mi.Color("black")))
Opacity of minor Tick line	Property:MinorTickLines.Opacity (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorTickLines(mit => mit.Opacity("0.5"))))	Not Applicable
Minor ticks per interval of primaryXAxis	Property:MinorTicksPerIntervalHtml.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MinorTicksPerInterval("4"))	Property:MinorTicksPerInterval@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MinorTicksPerInterval("4"))
name of the primaryXAxis	Property:Name (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Name("primaryXAxis")))	Property:Name@Html.EJS().Chart("container").PrimaryXAxis(px=>px.Name("primaryXAxis"))
Orientation of primaryXAxis	Property:Orientation (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Orientation("Horizontal")))	Not Applicable

Plot offset for primaryXAxis	Property:PlotOffset (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.PlotOffset("0")))	Property:PlotOffset@Html.EJS() .Chart("container").PrimaryXAxis(px=>px.PlotOffset("0"))
minimum for primaryXAxis	Property:Range.Minimum (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Range(r => r.Minimum("10"))))	Property:Minimum@Html.EJS() .Chart("container").PrimaryXAxis(px=>px.Minimum("10"))
maximum for primaryXAxis	Property:Range.Maximum (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Range(r => r.Maximum("23"))))	Property:Maximum@Html.EJS() .Chart("container").PrimaryXAxis(px=>px.Maximum("23"))
interval for primaryXAxis	Property:Range.Interval (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Range(r => r.Interval("2"))))	Property:Interval@Html.EJS() .Chart("container").PrimaryXAxis(px=>px.Interval("2"))
RangePadding for primaryXAxis	Property:RangePadding (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.RangePadding("none")))	Property:RangePadding@Html.EJS() .Chart("container").PrimaryXAxis(px=>px.RangePadding("none"))
Rounding Places in primaryXAxis	Property:RoundingPlaces (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.RoundingPlaces("3")))	Property:LabelFormat@Html.EJS() .Chart("container").PrimaryXAxis(px=>px.LabelFormat("n3"))
Scrollbar settings of primary	Property:ScrollbarSettings (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ScrollbarSettings()))	Not Applicable

Primary X Axis		
Tick Position in Primary X Axis	Property: TickLinesPosition (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.TickLinesPosition("Inside")))	Property: TickPosition@Html.EJS().Chart("container").PrimaryXAxis(px=>px.TickPosition("Inside"))
Value Type of Primary X Axis	Property: ValueType (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ValueType("DateTime")))	Property: ValueType@Html.EJS().Chart("container").PrimaryXAxis(px=>px.ValueType("DateTime"))
Visibility of Primary X Axis	Property: Visible (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Visible(true)))	Property: Visible@Html.EJS().Chart("container").PrimaryXAxis(px=>px.Visible(true))
Zoom Factor of Primary X Axis	Property: ZoomFactor (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ZoomFactor("0.3")))	Property: ZoomFactor@Html.EJS().Chart("container").PrimaryXAxis(px=>px.ZoomFactor("0.3"))
Zoom Position of Primary X Axis	Property: ZoomPosition (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.ZoomPosition("0.3")))	Property: ZoomPosition@Html.EJS().Chart("container").PrimaryXAxis(px=>px.ZoomPosition("0.3"))
Label Border of Primary X Axis	Property: LabelBorder (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.LabelBorder(lb => lb.Color("red").Width("2"))))	Property: Border@Html.EJS().Chart("container").PrimaryXAxis(px=>px.Border(b => b.Color("red").Width("2")))
Title of Primary X Axis	Property: Title.Text (Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.Title(t => t.Text("Chart Title"))))	Property: Title@Html.EJS().Chart("container").PrimaryXAxis(px=>px.Title("Chart Title"))

StripLine of primaryXAxis	<code>Property:StripLine(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.StripLine()))</code>	<code>Property:StripLines@Html.EJS().Chart("container").PrimaryXAxis(px=>px.StripLines())</code>
Multilevel labels of primaryXAxis	<code>Property:MultiLevelLabels(Html.EJ().Chart("chartContainer").PrimaryXAxis(px=>px.MultiLevelLabels()))</code>	<code>Property:MultiLevelLabels@Html.EJS().Chart("container").PrimaryXAxis(px=>px.MultiLevelLabels())</code>
skeleton for an axes	Not Applicable	<code>Property:Skeleton@Html.EJS().Chart("container").Axes(ax=>ax.Skeleton('yMd'))</code>
skeleton type for an axes	Not Applicable	<code>Property:SkeletonType@Html.EJS().Chart("container").Axes(ax=>ax.SkeletonType("DateTime"))</code>

primaryYAxis

Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	<code>Property:AlternateGridBand@(Html.EJ().Chart("chartContainer").PrimaryYAxis(px=>px.AlternateGridBand(alt => alt.even(e => e.Fill("red")))))</code>	Not applicable
Axis line cross value	<code>Property:CrossesAt@(Html.EJ().Chart("chartContainer").PrimaryYAxis(px=>px.CrossesAt(0)))</code>	<code>Property:CrossesAt@Html.EJS().Chart("container").PrimaryYAxis(px=>px.CrossesAt(0))</code>
Axis name with which the axis line has to be	<code>Property:CrossesInAxis@(Html.EJ().Chart("chartContainer").PrimaryYAxis(px=>px.CrossesInAxis("")))</code>	<code>Property:CrossesInAxis@Html.EJS().Chart("container").PrimaryYAxis(px=>px.CrossesInAxis(""))</code>

crossed		
Axis elements placed with axis line	<code>Property:ShowNextToAxisLine@(Html.EJ().Chart("chartContainer").PrimaryYAxis(px=>px.ShowNextToAxisLine(true)))</code>	<code>Property:PlaceNextToAxisLine@Html.EJS().Chart("container").primaryYAxis(px=>px.PlaceNextToAxisLine(true))</code>
Axis line style	<code>Property:AxisLine.Color@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.AxisLine(al => al.Color("red"))))</code>	<code>Property:LineStyle.Color@Html.EJS().Chart("container").primaryYAxis(px=>px.LineStyle(ls => ls.Color("black")))</code>
Axis line dasharray	<code>Property:AxisLine.DashArray@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.AxisLine(al => al.DashArray("10,5"))))</code>	<code>Property:LineStyle.DashArray@Html.EJS().Chart("container").primaryYAxis(px=>px.LineStyle(ls => ls.DashArray("10,5")))</code>
Offset for axis	<code>Property:AxisLine.Offset@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.AxisLine(al => al.Offset("10"))))</code>	<code>Property:PlotOffset@Html.EJS().Chart("container").primaryYAxis(px=>px.LineStyle(ls => ls.PlotOffset("10")))</code>
Visible of an axis	<code>Property:AxisLine.Visible@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.AxisLine(al => al.Visible(false))))</code>	<code>Property:Visible@Html.EJS().Chart("container").primaryYAxis(px=>px.LineStyle(ls => ls.Visible(false)))</code>
Width of an axis	<code>Property:AxisLine.Width@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.AxisLine(al => al.Width("2"))))</code>	<code>Property:LineStyle.Width@Html.EJS().Chart("container").primaryYAxis(px=>px.LineStyle(ls => ls.Width("2")))</code>
Column index of an axis	<code>Property:ColumnIndex@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.ColumnIndex("2")))</code>	<code>Property:ColumnIndex@Html.EJS().Chart("container").primaryYAxis(px=>px.ColumnIndex("2"))</code>
Span of an axis to place horizontally or vertically	<code>Property:ColumnSpan@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.ColumnSpan("2")))</code>	<code>Property:Span@Html.EJS().Chart("container").primaryYAxis(px=>px.Span("2"))</code>

Cross hair label of an axis	Property: CrossHairLabel.Visible (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.CrossHairLabel(cr => cr.Visible(true))))	Property: CrossHairTooltip.Enable@Html.EJS().Chart("container").primaryYAxis(px=>px.CrossHairLabel(cr => cr.Enable(true)))
Cross hair label color of an axis	Not applicable	Property: CrossHairTooltip.Fill@Html.EJS().Chart("container").primaryYAxis(px=>px.CrossHairLabel(cr => cr.Fill("red")))
Cross hair label text style	Not applicable	Property: CrossHairTooltip.TextStyle@Html.EJS().Chart("container").primaryYAxis(px=>px.CrossHairLabel(cr => cr.TextStyle()))
Desired interval count for primaryYAxis	Property: DesiredIntervals@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.DesiredIntervals("4")))	Property: DesiredIntervals@Html.EJS().Chart("container").primaryYAxis(px=>px.DesiredIntervals("4"))
Edges primaryYAxis	Property: EdgeLabelPlacement@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.EdgeLabelPlacement("none")))	Property: EdgeLabelPlacement@Html.EJS().Chart("container").primaryYAxis(px=>px.EdgeLabelPlacement("none"))
Enables trim for axis labels	Property: EnableTrim@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.EnableTrim(true)))	Property: EnableTrim@Html.EJS().Chart("container").primaryYAxis(px=>px.EnableTrim(true))
Specifies the interval of the axis according	Property: EnableAutoIntervalOnZooming@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.EnableAutoIntervalOnZooming(true)))	Property: EnableAutoIntervalOnZooming@Html.EJS().Chart("container").primaryYAxis(px=>px.EnableAutoIntervalOnZooming(true))

to the zoomed data of the chart		
Title style for primaryYAxis	<code>Property:Font@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Font()))</code>	<code>Property:TitleStyle@Html.EJS().Chart("container").primaryYAxis(px=>px.TitleStyle())</code>
Indexed for category axis	<code>Property:IsIndexed@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.IsIndexed(true)))</code>	<code>Property:IsIndexed@Html.EJS().Chart("container").primaryYAxis(px=>px.IsIndexed(true))</code>
Interval type for date time axis	<code>Property:IntervalType@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.IntervalType("Auto")))</code>	<code>Property:IntervalType@Html.EJS().Chart("container").primaryYAxis(px=>px.IntervalType("Auto"))</code>
Inversed axis	<code>Property:IsInversed@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.IsInversed(true)))</code>	<code>Property:IsInversed@Html.EJS().Chart("container").primaryYAxis(px=>px.IsInversed(true))</code>
Custom label format	<code>Property:LabelFormat@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LabelFormat('{value}K')))</code>	<code>Property:LabelFormat@Html.EJS().Chart("container").primaryYAxis(px=>px.LabelFormat('{value}K'))</code>
labelIntersectAction	<code>Property:LabelIntersectAction@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LabelIntersectAction("trim")))</code>	<code>Property:LabelIntersectAction@Html.EJS().Chart("container").primaryYAxis(px=>px.LabelIntersectAction("Trim"))</code>
labelPosition	<code>Property:LabelPosition@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LabelPosition("inside")))</code>	<code>Property:LabelPosition@Html.EJS().Chart("container").primaryYAxis(px=>px.LabelPosition("Inside"))</code>
labelPlacement for	<code>Property:LabelPlacement@(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LabelPlacement("onTicks")))</code>	<code>Property:LabelPlacement@Html.EJS().Chart("container").primaryYAxis(px=>px.LabelPlacement("onTicks"))</code>

category axis		
Axis label alignment	Property:Alignment@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Alignment("center")))	Not Applicable
Rotation of axis labels	Property:LabelRotation@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LabelRotation("45")))	Property:LabelRotation@Html.EJS().Chart("container").primaryYAxis(px=>px.LabelRotation("45"))
Log base value for logarithmic axis	Property:LogBase@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LogBase("10")))	Property:LogBase@Html.EJS().Chart("container").primaryYAxis(px=>px.LogBase("10"))
Major grid line visible	Property:MajorGridLines.visible@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Visible(true))))	Not Applicable
Width of Major GridLines	Property:MajorGridLines.Width@ (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Width("2"))))	Property:MajorGridLines.Width@Html.EJS().Chart("container").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Width("2")))
Color of Major GridLines	Property:MajorGridLines.Color (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Color("black"))))	Property:MajorGridLines.Color@Html.EJS().Chart("container").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Color("black")))
Dash Array of Major GridLines	Property:MajorGridLines.DashArray (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.DashArray("10,2"))))	Property:MajorGridLines.DashArray@Html.EJS().Chart("container").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.DashArray("10,2")))
Opacity of major	Property:MajorGridLines.Opacity (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Opacity("0.5"))))	Not Applicable

grid line		
Major Tick line	Property:MajorTickLines.Visible (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorTickLines(mtl => mtl.Visible(true))))	Not Applicable
Width of Major Tick Lines	Property:MajorTickLines.Width (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorTickLines(mtl => mtl.Width("2"))))	Property:MajorTickLines.Width@Html.EJS().Chart("container").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Width("2")))
Height of Major Tick Lines	Property:MajorTickLines.Size (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorTickLines(mtl => mtl.Size("2"))))	Property:MajorTickLines.Height@Html.EJS().Chart("container").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Height("2")))
Color of Major Tick Lines	Property:MajorTickLines.Color (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorTickLines(mtl => mtl.Color("black"))))	Property:MajorTickLines.Color@Html.EJS().Chart("container").primaryYAxis(px=>px.MajorGridLines(mgl => mgl.Color("black")))
Opacity of major Tick line	Property:MajorTickLines.Opacity (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MajorTickLines(mtl => mtl.Opacity("0.5"))))	Not Applicable
maximum labels of primaryYAxis	Property:MaximumLabels (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MaximumLabels("5")))	Property:MaximumLabels@Html.EJS().Chart("container").primaryYAxis(px=>px.MaximumLabels("5"))
maximum labels width of primaryYAxis is to trim	Property:MaximumLabelWidth (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MaximumLabelWidth("40")))	Property:MaximumLabelWidth@Html.EJS().Chart("container").primaryYAxis(px=>px.MaximumLabelWidth("40"))

minor grid line	Property:MinorGridLines.Visible (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorGridLines(mi =>mi.Visible(true))))	Not Applicable
Width of minor GridLines	Property:MinorGridLines.Width (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorGridLines(mi =>mi.Width("2"))))	Property:MinorGridLines.Width@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorGridLines(mi =>mi.Width("2")))
Color of minor GridLines	Property:MinorGridLines.Color (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorGridLines(mi =>mi.Color("black"))))	Property:MinorGridLines.Color@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorGridLines(mi =>mi.Color("black")))
Dash Array of minor GridLines	Property:MinorGridLines.DashArray (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorGridLines(mi =>mi.DashArray("10,5"))))	Property:MinorGridLines.DashArray@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorGridLines(mi =>mi.DashArray("10,5")))
Opacity of minor grid line	Property:MinorGridLines.Opacity (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorGridLines(mi =>mi.Opacity("0.5"))))	Not Applicable
minor Tick line	Property:MinorTickLines.Visible (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorTickLines(mit =>mit.Visible(true))))	Not Applicable
Width of minor TickLines	Property:MinorTickLines.Width (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorTickLines(mit =>mit.Width("2"))))	Property:MinorTickLines.Width@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorTickLines(mi =>mi.Width("2")))
Height of minor TickLines	Property:MinorTickLines.Size (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorTickLines(mit =>mit.Size("2"))))	Property:MinorTickLines.Height@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorTickLines(mi =>mi.Height("2")))
Color of minor	Property:MinorTickLines.Color(Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorTickLines(mit =>mit.Color("black"))))	Property:MinorTickLines.Color@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorTickLines(mi =>mi.Color("black")))

TickLines		
Opacity of minor Tick line	Property:MinorTickLines.Opacity (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorTickLines(mit => mit.Opacity("0.5"))))	Not Applicable
Minor ticks per interval of primaryYAxis	Property:MinorTicksPerIntervalHtml.EJ().Chart("chartContainer").primaryYAxis(px=>px.MinorTicksPerInterval("4"))	Property:MinorTicksPerInterval@Html.EJS().Chart("container").primaryYAxis(px=>px.MinorTicksPerInterval("4"))
name of the primaryYAxis	Property:Name (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Name("primaryYAxis")))	Property:Name@Html.EJS().Chart("container").primaryYAxis(px=>px.Name("primaryYAxis"))
Orientation of primaryYAxis	Property:Orientation (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Orientation("Horizontal")))	Not Applicable
Plot offset for primaryYAxis	Property:PlotOffset (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.PlotOffset("0")))	Property:PlotOffset@Html.EJS().Chart("container").primaryYAxis(px=>px.PlotOffset("0"))
minimum for primaryYAxis	Property:Range.Minimum (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Range(r => r.Minimum("10"))))	Property:Minimum@Html.EJS().Chart("container").primaryYAxis(px=>px.Minimum("10"))
maximum for primaryYAxis	Property:Range.Maximum (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Range(r => r.Maximum("23"))))	Property:Maximum@Html.EJS().Chart("container").primaryYAxis(px=>px.Maximum("23"))

primaryYAxis		
interval for primaryYAxis	Property:Range.Interval (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Range(r => r.Interval("2"))))	Property:Interval@Html.EJS() .Chart("container").primaryYAxis(px=>px.Interval("2"))
RangePadding for primaryYAxis	Property:RangePadding (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.RangePadding("none")))	Property:RangePadding@Html.EJS() .Chart("container").primaryYAxis(px=>px.RangePadding("none"))
Rounding Places in primaryYAxis	Property:RoundingPlaces (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.RoundingPlaces("3")))	Property:LabelFormat@Html.EJS() .Chart("container").primaryYAxis(px=>px.LabelFormat("n3"))
Scrollbar settings of primaryYAxis	Property:ScrollbarSettings (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.ScrollbarSettings()))	Not Applicable
TickPosition in primaryYAxis	Property:TickLinesPosition (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.TickLinesPosition("Inside")))	Property:TickPosition@Html.EJS() .Chart("container").primaryYAxis(px=>px.TickPosition("Inside"))
value Type of primaryYAxis	Property:ValueType (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.ValueType("DateTime")))	Property:ValueType@Html.EJS() .Chart("container").primaryYAxis(px=>px.ValueType("DateTime"))
visibility of primaryYAxis	Property:Visible (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Visible(true)))	Property:Visible@Html.EJS() .Chart("container").primaryYAxis(px=>px.Visible(true))

primaryYAxis		
zoomFactor of primaryYAxis	<code>Property:ZoomFactor (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.ZoomFactor("0.3")))</code>	<code>Property:ZoomFactor@Html.EJS().Chart("container").primaryYAxis(px=>px.ZoomFactor("0.3"))</code>
zoomPosition of primaryYAxis	<code>Property:ZoomPosition (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.ZoomPosition("0.3")))</code>	<code>Property:ZoomPosition@Html.EJS().Chart("container").primaryYAxis(px=>px.ZoomPosition("0.3"))</code>
labelBorder of primaryYAxis	<code>Property:LabelBorder (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.LabelBorder(lb => lb.Color("red").Width("2"))))</code>	<code>Property:Border@Html.EJS().Chart("container").primaryYAxis(px=>px.Border(b => b.Color("red").Width("2")))</code>
title of primaryYAxis	<code>Property:Title.Text (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.Title(t => t.Text("Chart Title"))))</code>	<code>Property:Title@Html.EJS().Chart("container").primaryYAxis(px=>px.Title("Chart Title"))</code>
StripLine of primaryYAxis	<code>Property:StripLine (Html.EJ().Chart("chartContainer").primaryYAxis(px=>px.StripLine()))</code>	<code>Property:StripLines@Html.EJS().Chart("container").primaryYAxis(px=>px.StripLines())</code>
Multilevel labels of primaryYAxis	<code>Property:MultiLevelLabels (Html.EJ().Chart("chartContainer").PrimaryYAxis(px=>px.MultiLevelLabels))</code>	<code>Property:MultiLevelLabels@Html.EJS().Chart("container").PrimaryYAxis(px=>px.MultiLevelLabels())</code>
skeleton for axes	Not Applicable	<code>Property:Skeleton@Html.EJS().Chart("container").Axes(ax=>ax.Skeleton('yMd'))</code>

skeleton type for an axes	Not Applicable	Property:SkeletonType@Html.EJS().Chart("container").Axes(ax=>ax.SkeletonType("DateTime"))
---------------------------	----------------	---

Axes

Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	Property:AlternateGridBand@(Html.EJ().Chart("chartContainer").Axes(px=>px.AlternateGridBand(alt => alt.even(e => e.Fill("red")))))	Not applicable
Axis line cross value	Property:CrossesAt@(Html.EJ().Chart("chartContainer").Axes(px=>px.CrossesAt(0)))	Property:CrossesAt@Html.EJS().Chart("container").Axes(px=>px.CrossesAt(0))
Axis name with which the axis line has to be crossed	Property:CrossesInAxis@(Html.EJ().Chart("chartContainer").Axes(px=>px.CrossesInAxis("")))	Property:CrossesInAxis@Html.EJS().Chart("container").Axes(px=>px.CrossesInAxis(""))
Axis elements placed with axis line	Property:ShowNextToAxisLine@(Html.EJ().Chart("chartContainer").Axes(px=>px.ShowNextToAxisLine(true)))	Property:PlaceNextToAxisLine@Html.EJS().Chart("container").Axes(px=>px.PlaceNextToAxisLine(true))
Axis line style	Property:AxisLine.Color@(Html.EJ().Chart("chartContainer").Axes(px=>px.AxisLine(al => al.Color("red"))))	Property:LineStyle.Color@Html.EJS().Chart("container").Axes(px=>px.LineStyle(ls => ls.Color("black")))
Axis line dashArray	Property:AxisLine.DashArray@(Html.EJ().Chart("chartContainer").Axes(px=>px.AxisLine(al => al.DashArray("10,5"))))	Property:LineStyle.DashArray@Html.EJS().Chart("container").Axes(px=>px.LineStyle(ls => ls.DashArray("10,5")))

Offset for axis	Property:AxisLine.Offset@(Html.EJ().Chart("chartContainer").Axes(px=>px.AxisLine(al => al.Offset("10"))))	Property:PlotOffset@Html.EJS().Chart("container").Axes(px=>px.LineStyle(ls => ls.PlotOffset("10")))
Visible of an axis	Property:AxisLine.Visible@(Html.EJ().Chart("chartContainer").Axes(px=>px.AxisLine(al => al.Visible(false))))	Property:Visible@Html.EJS().Chart("container").Axes(px=>px.LineStyle(ls => ls.Visible(false)))
Width of an axis	Property:AxisLine.Width@(Html.EJ().Chart("chartContainer").Axes(px=>px.AxisLine(al => al.Width("2"))))	Property:LineStyle.Width@Html.EJS().Chart("container").Axes(px=>px.LineStyle(ls => ls.Width("2")))
Column index of an axis	Property:ColumnIndex@(Html.EJ().Chart("chartContainer").Axes(px=>px.ColumnIndex("2")))	Property:ColumnIndex@Html.EJS().Chart("container").Axes(px=>px.ColumnIndex("2"))
Span of an axis to place horizontally or vertically	Property:ColumnSpan@(Html.EJ().Chart("chartContainer").Axes(px=>px.ColumnSpan("2")))	Property:Span@Html.EJS().Chart("container").Axes(px=>px.Span("2"))
Cross hair label of an axis	Property:CrossHairLabel.Visible(Html.EJ().Chart("chartContainer").Axes(px=>px.CrossHairLabel(cr => cr.Visible(true))))	Property:CrossHairTooltip.Enable@Html.EJS().Chart("container").Axes(px=>px.CrossHairLabel(cr => cr.Enable(true)))
Cross hair label color of an axis	Not applicable	Property:CrossHairTooltip.Fill@Html.EJS().Chart("container").Axes(px=>px.CrossHairLabel(cr => cr.Fill("red")))
Cross hair label text style	Not applicable	Property:CrossHairTooltip.TextStyle@Html.EJS().Chart("container").Axes(px=>px.CrossHairLabel(cr => cr.TextStyle()))
Desired interval	Property:DesiredIntervals@(Html.EJ().Chart("chartContainer").Axes(px=>px.DesiredIntervals("4")))	Property:DesiredIntervals@Html.EJS().Chart("container").Axes(px=>px.DesiredIntervals("4"))

count for Axes		
Edges Axes	<code>Property:EdgeLabelPlacement@(Html.EJ().Chart("chartContainer").Axes(px=>px.EdgeLabelPlacement("none")))</code>	<code>Property:EdgeLabelPlacement@Html.EJS().Chart("container").Axes(px=>px.EdgeLabelPlacement("none"))</code>
Enables trim for axis labels	<code>Property:EnableTrim@(Html.EJ().Chart("chartContainer").Axes(px=>px.EnableTrim(true)))</code>	<code>Property:EnableTrim@Html.EJS().Chart("container").Axes(px=>px.EnableTrim(true))</code>
Specifies the interval of the axis according to the zoomed data of the chart	<code>Property:EnableAutoIntervalOnZooming@(Html.EJ().Chart("chartContainer").Axes(px=>px.EnableAutoIntervalOnZooming(true)))</code>	<code>Property:EnableAutoIntervalOnZooming@Html.EJS().Chart("container").Axes(px=>px.EnableAutoIntervalOnZooming(true))</code>
Title style for Axes	<code>Property:Font@(Html.EJ().Chart("chartContainer").Axes(px=>px.Font()))</code>	<code>Property:TitleStyle@Html.EJS().Chart("container").Axes(px=>px.TitleStyle())</code>
Indexed for category axis	<code>Property:IsIndexed@(Html.EJ().Chart("chartContainer").Axes(px=>px.IsIndexed(true)))</code>	<code>Property:IsIndexed@Html.EJS().Chart("container").Axes(px=>px.IsIndexed(true))</code>
Interval type for date time axis	<code>Property:IntervalType@(Html.EJ().Chart("chartContainer").Axes(px=>px.IntervalType("Auto")))</code>	<code>Property:IntervalType@Html.EJS().Chart("container").Axes(px=>px.IntervalType("Auto"))</code>

Inversed axis	<code>Property:IsInversed@(Html.EJ().Chart("chartContainer").Axes(px=>px.IsInversed(true)))</code>	<code>Property:IsInversed@Html.EJS().Chart("container").Axes(px=>px.IsInversed(true))</code>
Custom label format	<code>Property:LabelFormat@(Html.EJ().Chart("chartContainer").Axes(px=>px.LabelFormat('{value}K')))</code>	<code>Property:LabelFormat@Html.EJS().Chart("container").Axes(px=>px.LabelFormat('{value}K'))</code>
labelIntersectAction	<code>Property:LabelIntersectAction@(Html.EJ().Chart("chartContainer").Axes(px=>px.LabelIntersectAction("trim")))</code>	<code>Property:LabelIntersectAction@Html.EJS().Chart("container").Axes(px=>px.LabelIntersectAction("Trim"))</code>
labelPosition	<code>Property:LabelPosition@(Html.EJ().Chart("chartContainer").Axes(px=>px.LabelPosition("inside")))</code>	<code>Property:LabelPosition@Html.EJS().Chart("container").Axes(px=>px.LabelPosition("Inside"))</code>
labelPlacement for category axis	<code>Property:LabelPlacement@(Html.EJ().Chart("chartContainer").Axes(px=>px.LabelPlacement("onTicks")))</code>	<code>Property:LabelPlacement@Html.EJS().Chart("container").Axes(px=>px.LabelPlacement("onTicks"))</code>
Axis label alignment	<code>Property:Alignment@(Html.EJ().Chart("chartContainer").Axes(px=>px.Alignment("center")))</code>	Not Applicable
Rotation of axis labels	<code>Property:LabelRotation@(Html.EJ().Chart("chartContainer").Axes(px=>px.LabelRotation("45")))</code>	<code>Property:LabelRotation@Html.EJS().Chart("container").Axes(px=>px.LabelRotation("45"))</code>
Log base value for logarithmic axis	<code>Property:LogBase@(Html.EJ().Chart("chartContainer").Axes(px=>px.LogBase("10")))</code>	<code>Property:LogBase@Html.EJS().Chart("container").Axes(px=>px.LogBase("10"))</code>
Major grid line visible	<code>Property:MajorGridLines.visible@(Html.EJ().Chart("chartContainer").Axes(px=>px.MajorGridLines(mgl => mgl.Visible(true)))</code>	Not Applicable

Width of Major GridLines	Property:MajorGridLines.Width@ (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorGridLines (mgl => mgl.Width ("2"))))	Property:MajorGridLines.Width@Html.EJS() .Chart ("container") .Axes (px=>px.MajorGridLines (mgl => mgl.Width ("2")))
Color of Major GridLines	Property:MajorGridLines.Color (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorGridLines (mgl => mgl.Color ("black"))))	Property:MajorGridLines.Color@Html.EJS() .Chart ("container") .Axes (px=>px.MajorGridLines (mgl => mgl.Color ("black")))
DashArray of Major GridLines	Property:MajorGridLines.DashArray (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorGridLines (mgl => mgl.DashArray ("10,2"))))	Property:MajorGridLines.DashArray@Html.EJS() .Chart ("container") .Axes (px=>px.MajorGridLines (mgl => mgl.DashArray ("10,2")))
Opacity of major grid line	Property:MajorGridLines.Opacity (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorGridLines (mgl => mgl.Opacity ("0.5"))))	Not Applicable
Major Tick line	Property:MajorTickLines.Visible (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorTickLines (mtl => mtl.Visible (true))))	Not Applicable
Width of Major TickLines	Property:MajorTickLines.Width (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorTickLines (mtl => mtl.Width ("2"))))	Property:MajorTickLines.Width@Html.EJS() .Chart ("container") .Axes (px=>px.MajorGridLines (mgl => mgl.Width ("2")))
Height of Major TickLines	Property:MajorTickLines.Size (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorTickLines (mtl => mtl.Size ("2"))))	Property:MajorTickLines.Height@Html.EJS() .Chart ("container") .Axes (px=>px.MajorGridLines (mgl => mgl.Height ("2")))
Color of Major TickLines	Property:MajorTickLines.Color (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorTickLines (mtl => mtl.Color ("black"))))	Property:MajorTickLines.Color@Html.EJS() .Chart ("container") .Axes (px=>px.MajorGridLines (mgl => mgl.Color ("black")))
Opacity of major	Property:MajorTickLines.Opacity (Html.EJ() .Chart ("chartContainer") .Axes (px=>px.MajorTickLines (mtl => mtl.Opacity ("0.5"))))	Not Applicable

Tick line	<code>jorTickLines (mtl => mtl.Opacity("0.5"))))</code>	
maximum labels of Axes	<code>Property:MaximumLabels (Html.EJ().Chart("chartContainer").Axes(px=>px.MaximumLabels("5")))</code>	<code>Property:MaximumLabels@Html.EJS().Chart("container").Axes(px=>px.MaximumLabels("5"))</code>
maximum labels width of Axes to trim	<code>Property:MaximumLabelWidth (Html.EJ().Chart("chartContainer").Axes(px=>px.MaximumLabelWidth("40")))</code>	<code>Property:MaximumLabelWidth@Html.EJS().Chart("container").Axes(px=>px.MaximumLabelWidth("40"))</code>
minor grid line	<code>Property:MinorGridLines.Visible (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorGridLines(mi => mi.Visible(true))))</code>	Not Applicable
Width of minor GridLines	<code>Property:MinorGridLines.Width (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorGridLines(mi => mi.Width("2"))))</code>	<code>Property:MinorGridLines.Width@Html.EJS().Chart("container").Axes(px=>px.MinorGridLines(mi => mi.Width("2")))</code>
Color of minor GridLines	<code>Property:MinorGridLines.Color (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorGridLines(mi => mi.Color("black"))))</code>	<code>Property:MinorGridLines.Color@Html.EJS().Chart("container").Axes(px=>px.MinorGridLines(mi => mi.Color("black")))</code>
DashArray of minor GridLines	<code>Property:MinorGridLines.DashArray (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorGridLines(mi => mi.DashArray("10,5"))))</code>	<code>Property:MinorGridLines.DashArray@Html.EJS().Chart("container").Axes(px=>px.MinorGridLines(mi => mi.DashArray("10,5")))</code>
Opacity of minor grid line	<code>Property:MinorGridLines.Opacity (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorGridLines(mi => mi.Opacity("0.5"))))</code>	Not Applicable
minor Tick line	<code>Property:MinorTickLines.Visible (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorTickLines(mit => mit.Visible(true))))</code>	Not Applicable

Width of minor TickLines	Property:MinorTickLines.Width (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorTickLines(mit => mit.Width("2"))))	Property:MinorTickLines.Width@Html.EJS().Chart("container").Axes(px=>px.MinorTickLines(mi => mi.Width("2")))
Height of minor TickLines	Property:MinorTickLines.Size (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorTickLines(mit => mit.Size("2"))))	Property:MinorTickLines.Height@Html.EJS().Chart("container").Axes(px=>px.MinorTickLines(mi => mi.Height("2")))
Color of minor TickLines	Property:MinorTickLines.Color(Html.EJ().Chart("chartContainer").Axes(px=>px.MinorTickLines(mit => mit.Color("black"))))	Property:MinorTickLines.Color@Html.EJS().Chart("container").Axes(px=>px.MinorTickLines(mi => mi.Color("black")))
Opacity of minor Tick line	Property:MinorTickLines.Opacity (Html.EJ().Chart("chartContainer").Axes(px=>px.MinorTickLines(mit => mit.Opacity("0.5"))))	Not Applicable
Minor ticks per interval of Axes	Property:MinorTicksPerIntervalHtml.EJ().Chart("chartContainer").Axes(px=>px.MinorTicksPerInterval("4"))	Property:MinorTicksPerInterval@Html.EJS().Chart("container").Axes(px=>px.MinorTicksPerInterval("4"))
name of the Axes	Property:Name (Html.EJ().Chart("chartContainer").Axes(px=>px.Name("Axes")))	Property:Name@Html.EJS().Chart("container").Axes(px=>px.Name("Axes"))
Orientation of Axes	Property:Orientation (Html.EJ().Chart("chartContainer").Axes(px=>px.Orientation("Horizontal")))	Not Applicable
Plot offset for Axes	Property:PlotOffset (Html.EJ().Chart("chartContainer").Axes(px=>px.PlotOffset("0")))	Property:PlotOffset@Html.EJS().Chart("container").Axes(px=>px.PlotOffset("0"))
minimum for Axes	Property:Range.Minimum (Html.EJ().Chart("chartContainer").Axes(px=>px.Range(r => r.Minimum("10"))))	Property:Minimum@Html.EJS().Chart("container").Axes(px=>px.Minimum("10"))

maximum for Axes	Property: Range.Maximum (Html.EJ().Chart("chartContainer").Axes(px=>px.Range(r=> r.Maximum("23"))))	Property: Maximum@Html.EJS().Chart("container").Axes(px=>px.Maximum("23"))
interval for Axes	Property: Range.Interval (Html.EJ().Chart("chartContainer").Axes(px=>px.Range(r=> r.Interval("2"))))	Property: Interval@Html.EJS().Chart("container").Axes(px=>px.Interval("2"))
Range Padding for Axes	Property: RangePadding (Html.EJ().Chart("chartContainer").Axes(px=>px.RangePadding("none")))	Property: RangePadding@Html.EJS().Chart("container").Axes(px=>px.RangePadding("none"))
Rounding Places in Axes	Property: RoundingPlaces (Html.EJ().Chart("chartContainer").Axes(px=>px.RoundingPlaces("3")))	Property: LabelFormat@Html.EJS().Chart("container").Axes(px=>px.LabelFormat("n3"))
Scrollbar settings of Axes	Property: ScrollbarSettings (Html.EJ().Chart("chartContainer").Axes(px=>px.ScrollbarSettings()))	Not Applicable
Tick Position in Axes	Property: TickLinesPosition (Html.EJ().Chart("chartContainer").Axes(px=>px.TickLinesPosition("Inside")))	Property: TickPosition@Html.EJS().Chart("container").Axes(px=>px.TickPosition("Inside"))
valueType of Axes	Property: ValueType (Html.EJ().Chart("chartContainer").Axes(px=>px.ValueType("DateTime")))	Property: ValueType@Html.EJS().Chart("container").Axes(px=>px.ValueType("DateTime"))
visible of Axes	Property: Visible (Html.EJ().Chart("chartContainer").Axes(px=>px.Visible(true)))	Property: Visible@Html.EJS().Chart("container").Axes(px=>px.Visible(true))
zoom Factor of Axes	Property: ZoomFactor (Html.EJ().Chart("chartContainer").Axes(px=>px.ZoomFactor("0.3")))	Property: ZoomFactor@Html.EJS().Chart("container").Axes(px=>px.ZoomFactor("0.3"))
zoom Position of Axes	Property: ZoomPosition (Html.EJ().Chart("chartContainer").Axes(px=>px.ZoomPosition("0.3")))	Property: ZoomPosition@Html.EJS().Chart("container").Axes(px=>px.ZoomPosition("0.3"))

labelB order of Axes	Property:LabelBorder @(Html.EJ().Chart("chartContainer").Axes(px=>px.LabelBorder(lb => lb.Color("red").Width("2"))))	Property:Border @Html.EJS().Chart("container").Axes(px=>px.Border(b => b.Color("red").Width("2")))
title of Axes	Property:Title.Text (Html.EJ().Chart("chartContainer").Axes(px=>px.Title(t => t.Text("Chart Title"))))	Property:Title @Html.EJS().Chart("container").Axes(px=>px.Title("Chart Title"))
StripLi ne of Axes	Property:StripLine (Html.EJ().Chart("chartContainer").Axes(px=>px.StripLine()))	Property:StripLines @Html.EJS().Chart("container").Axes(px=>px.StripLines())
Multil evel labels of Axes	Property:MultiLevelLabels (Html.EJ().Chart("chartContainer").Axes(px=>px.MultiLevelLabels()))	Property:MultiLevelLabels @Html.EJS().Chart("container").Axes(px=>px.MultiLevelLabels())
skelet on for an axes	Not Applicable	Property:Skeleton @Html.EJS().Chart("container").Axes(ax=>ax.Skeleton('yMd'))
skelet on type for an axes	Not Applicable	Property:SkeletonType @Html.EJS().Chart("container").Axes(ax=>ax.SkeletonType("DateTime"))

Rows

Behav iour	API in Essential JS 1	API in Essential JS 2
rows in chart	Property:RowDefinitions @(Html.EJ().Chart("chartContainer").RowDefinitions().Add();)	Property:Rows @Html.EJS().Chart("container").Rows().Add();
unit	Property:Unit @(Html.EJ().Chart("chartContainer").RowDefinitions(r => r.Unit("percentage")).Add();)	Not Applicable
height of rows in chart	Property:RowHeight @(Html.EJ().Chart("chartContainer").RowDefinitions(r => r.owHeight("50")).Add();)	Property:Height @Html.EJS().Chart("container").Rows(r => r.Height("300")).Add();

Line customization	Property:lineColor, lineWidth@ (Html.EJ().Chart("chartContainer").RowDefinitions(r => r.rowHeight("50").LineColor("brown").LineWidth("2")).Add());	Property:Border@Html.EJS().Chart("container").Rows({r => r.Border(b => b.Width("2").Color("brown").Height("300")).Add();});chart.appendTo('#chart');
--------------------	---	--

Series

Behavior	API in Essential JS 1	API in Essential JS 2
BearFillColor	Property: BearFillColor@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.BearFillColor("red").Add();}))	Property: BearFillColor@Html.EJS().Chart("container").Series(sr => {sr.BearFillColor("red").Add();}))
Border	Property: Border@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Border(br => br.Color("red").Width("2").DashArray("10,5")).Add();}))	Property: Border@Html.EJS().Chart("container").Series(sr => {sr.Border(br => br.Color("red").Width("2").DashArray("10,5")).Add();}))
BoxPlot Mode	Property: BoxPlotMode@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.BoxPlotMode("inclusive").Add();}))	Property: BoxPlotMode@Html.EJS().Chart("container").Series(sr => {sr.BoxPlotMode("inclusive").Add();}))
Minimum radius of Bubble series	Property: BubbleOptions.MinRadius@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.BubbleOptions(b => b.MinRadius("2")).Add();}))	Property: MinRadius@Html.EJS().Chart("container").Series(sr => {sr.BubbleOptions(b => b.MinRadius("2")).Add();}))
Maximum radius of Bubble series	Property: BubbleOptions.MaxRadius@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.BubbleOptions(b => b.MaxRadius("10")).Add();}))	Property: MaxRadius@Html.EJS().Chart("container").Series(sr => {sr.BubbleOptions(b => b.MaxRadius("10")).Add();}))
bullFillColor	Property: BullFillColor@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.BullFillColor("red").Add();}))	Property: BullFillColor@Html.EJS().Chart("container").Series(sr => {sr.BullFillColor("red").Add();}))
Cardinal spline tension for spline series	Property: CardinalSplineTension@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.CardinalSplineTension("0.5").Add();}))	Property: CardinalSplineTension@Html.EJS().Chart("container").Series(sr => {sr.CardinalSplineTension("0.5").Add();}))
Column Width for	Property: ColumnWidth@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.ColumnWidth("0.5").Add();}))	Property: ColumnWidth@Html.EJS().Chart("container").Series(sr => {sr.ColumnWidth("0.5").Add();}))

rectangle series		
Column spacing for rectangle series	<code>Property:ColumnSpacing@(Html.EJ().Chart("chartContainer").Series(sr => {sr.ColumnSpacing("0.5").Add();}))</code>	<code>Property:ColumnSpacing@Html.EJS().Chart("container").Series(sr => {sr.ColumnSpacing("0.5").Add();}))</code>
Topleft radius for rectangle series	<code>Property:CornerRadius.TopLeft@(Html.EJ().Chart("chartContainer").Series(sr => {sr.CornerRadius(c => c.TopLeft(0)).Add();}))</code>	<code>Property:CornerRadius.TopLeft@Html.EJS().Chart("container").Series(sr => {sr.CornerRadius(c => c.TopLeft(0)).Add();}))</code>
topRight radius for rectangle series	<code>Property:CornerRadius.TopRight@(Html.EJ().Chart("chartContainer").Series(sr => {sr.CornerRadius(c => c.TopRight(0)).Add();}))</code>	<code>Property:CornerRadius.TopRight@Html.EJS().Chart("container").Series(sr => {sr.CornerRadius(c => c.TopRight(0)).Add();}))</code>
bottomRight radius for rectangle series	<code>Property:CornerRadius.BottomRight@(Html.EJ().Chart("chartContainer").Series(sr => {sr.CornerRadius(c => c.BottomRight(0)).Add();}))</code>	<code>Property:CornerRadius.BottomRight@Html.EJS().Chart("container").Series(sr => {sr.CornerRadius(c => c.BottomRight(0)).Add();}))</code>
bottomLeft radius for rectangle series	<code>Property:CornerRadius.BottomLeft@(Html.EJ().Chart("chartContainer").Series(sr => {sr.CornerRadius(c => c.BottomLeft(0)).Add();}))</code>	<code>Property:CornerRadius.BottomLeft@Html.EJS().Chart("container").Series(sr => {sr.CornerRadius(c => c.BottomLeft(0)).Add();}))</code>
DashArray property	<code>Property:DashArray@(Html.EJ().Chart("chartContainer").Series(sr => {sr.DashArray("10,5").Add();}))</code>	<code>Property:DashArray@Html.EJS().Chart("container").Series(sr => {sr.DashArray("10,5").Add();}))</code>
DataSource for series	<code>Property:DataSource@(Html.EJ().Chart("chartContainer").Series(sr => {sr.DataSource().Add();}))</code>	<code>Property:DataSource@Html.EJS().Chart("container").Series(sr => {sr.DataSource().Add();}))</code>
Draw type for Polar series	<code>Property:DrawType@(Html.EJ().Chart("chartContainer").Series(sr => {sr.DrawType("Line").Add();}))</code>	<code>Property:DrawType@Html.EJS().Chart("container").Series(sr => {sr.DrawType("Line").Add();}))</code>
EmptyPointSettings	<code>Property:EmptyPointSettings.Visible@(Html.EJ().Chart("chartContainer").Series(sr</code>	Not Applicable

gs for series	<code>=> {sr.EmptyPointSettings(e => e.Visible(false)).Add();})</code>	
Empty Point Display mode	<code>Property:EmptyPointSettings.DisplayMode@(Html.EJ().Chart("chartContainer").Series(sr => {sr.EmptyPointSettings(e => e.DisplayMode("Average")).Add();})</code>	<code>Property:EmptyPointSettings.DisplayMode@Html.EJS().Chart("container").Series(sr => {sr.EmptyPointSettings(e => e.DisplayMode("Average")).Add();})</code>
Empty Point color	<code>Property:EmptyPointSettings.Color@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Color("red").Add();})</code>	<code>Property:EmptyPointSettings.Fill@Html.EJS().Chart("container").Series(sr => {sr.Fill("red").Add();})</code>
Empty Point Border	<code>Property:EmptyPointSettings.Border@(Html.EJ().Chart("chartContainer").Series(sr => {sr.EmptyPointSettings(e => e.Border(b => b.Color("red").Width("2"))).Add();})</code>	<code>Property:EmptyPointSettings.Border@Html.EJS().Chart("container").Series(sr => {sr.EmptyPointSettings(e => e.Border(b => b.Color("red").Width("2"))).Add();})</code>
Enable animation for series	<code>Property:EnableAnimation@(Html.EJ().Chart("chartContainer").Series(sr => {sr.EnableAnimation(true).Add();})</code>	<code>Property:Animation.Enable@Html.EJS().Chart("container").Series(sr => {sr.Animation(a => a.Enable(false)).Add();})</code>
Animation duration for series	<code>Property:AnimationDuration@(Html.EJ().Chart("chartContainer").Series(sr => {sr.AnimationDuration("1000").Add();})</code>	<code>Property:Animation.Duration@Html.EJS().Chart("container").Series(sr => {sr.Animation(a => a.Duration("1000").Add();})</code>
Animation delay for series	Not Applicable	<code>Property:Animation.Delay@Html.EJS().Chart("container").Series(sr => {sr.Animation(a => a.Delay("100").Add();})</code>
Drag settings for series	<code>Property:DragSettings@(Html.EJ().Chart("chartContainer").Series(sr => {sr.DragSettings(d => d.Mode("X")).Add();})</code>	Not Applicable
Errorbar settings for series	<code>Property:ErrorBarSettings@(Html.EJ().Chart("chartContainer").Series(sr => {sr.ErrorBarSettings().Add();})</code>	<code>Property:ErrorBarSettings@Html.EJS().Chart("container").Series(sr => {sr.ErrorBarSettings().Add();})</code>
Closed series	<code>Property:IsClosed@(Html.EJ().Chart("chartContainer").Series(sr => {sr.IsClosed(true).Add();})</code>	<code>Property:IsClosed@Html.EJS().Chart("container").Series(sr => {sr.IsClosed(true).Add();})</code>
Stacking Property for series	<code>Property:IsStacking@(Html.EJ().Chart("chartContainer").Series(sr => {sr.IsStacking(true).Add();})</code>	Not Applicable

Line cap for series	Property: LineCap@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.LineCap("butt").Add();}))	Not Applicable
Line join for series	Property: LineJoin@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.LineJoin("round").Add();}))	Not Applicable
Opacity for series	Property: Opacity@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Opacity("0.7").Add();}))	Property: Opacity@Html.EJS().Chart("container").Series(sr => {sr.Opacity("0.7").Add();}))
Outlier settings of series	Property: OutLierSettings@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.OutLierSettings(out => out.Shape("rectangle").Size(s => s.Height("30").Width("20"))).Add();}))	Not Applicable
Palette	Property: Palette@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Palette("ColorFieldName").Add();}))	Property: PointColorMapping@Html.EJS().Chart("container").Series(sr => {sr.PointColorMapping("color").Add();}))
Positive fill for waterfall series	Property: PositiveFill@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.PositiveFill("red").Add();}))	Property: PositiveFill@Html.EJS().Chart("container").Series(sr => {sr.PositiveFill("red").Add();}))
Show average value in box and whisker series	Property: ShowMedian@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.ShowMedian(true).Add();}))	Property: ShowMedian@Html.EJS().Chart("container").Series(sr => {sr.ShowMedian(true).Add();}))
To group the series of stacking collection.	Property: StackingGroup@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.StackingGroup("group").Add();}))	Property: StackingGroup@Html.EJS().Chart("container").Series(sr => {sr.StackingGroup("group").Add();}))
Specifies the type of the series to render in chart.	Property: Type@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Type(SeriesType.Line).Add();}))	Property: Type@Html.EJS().Chart("container").Series(sr => {sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Add();}))
Defines the visibility	Property: Visibility@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Visibility(true).Add();}))	Property: Visible@Html.EJS().Chart("container").Series(sr => {sr.Visible(true).Add();}))

of the series.		
Enables or disables the visibility of legend item.	Property:VisibleOnLegend <code>@(Html.EJ().Chart("chartContainer").Series(sr => {sr.VisibleOnLegend(true).Add();}))</code>	Property:ToggleVisibility @Html.EJS().Chart("container").LegendSettings(l => l.ToggleVisibility(true))})
Specifies the different types of spline curve.	Property:SplineType @(Html.EJ().Chart("chartContainer").Series(sr => {sr.SplineType('Natural').Add();}))	Property:SplineType @Html.EJS().Chart("container").Series(sr => {sr.SplineType('Natural').Add();})
Specifies the name of the x-axis that has to be associated with this series. Add an axis instance with this name to axes collection.	Property:XAxisName @(Html.EJ().Chart("chartContainer").Series(sr => {sr.XAxisName('secondaryXAxis').Add();}))	Property:XAxisName @Html.EJS().Chart("container").Series(sr => {sr.XAxisName('secondaryXAxis').Add();})
Name of the property in the datasource that contains x value for the series.	Property:XName @(Html.EJ().Chart("chartContainer").Series(sr => {sr.XName("x").Add();}))	Property:XName @Html.EJS().Chart("container").Series(sr => {sr.XName("x").Add();})

Specifies the name of the y-axis that has to be associated with this series. Add an axis instance with this name to axes collection.	Property:YAxisName @(Html.EJ().Chart("chartContainer").Series(sr => {sr.YAxisName("secondaryYAxis").Add();}))	Property:YAxisName @Html.EJS().Chart("container").Series(sr => {sr.YAxisName("secondaryYAxis").Add();}))
Name of the property in the datasource that contains y value for the series.	Property:YName @(Html.EJ().Chart("chartContainer").Series(sr => {sr.YName("Y").Add();}))	Property:YName @Html.EJS().Chart("container").Series(sr => {sr.YName("Y").Add();}))
Name of the property in the datasource that contains high value for the series.	Property:High @(Html.EJ().Chart("chartContainer").Series(sr => {sr.High("y").Add();}))	Property:High @Html.EJS().Chart("container").Series(sr => {sr.High("y").Add();}))
Name of the property in the datasource that contains	Property:Low @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Low("y").Add();}))	Property:Low @Html.EJS().Chart("container").Series(sr => {sr.Low("y").Add();}))

low value for the series.		
Name of the property in the datasource that contains close value for the series.	<code>Property:Close@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Close("y").Add();}))</code>	<code>Property:Close@Html.EJS().Chart("container").Series(sr => {sr.Close("y").Add();}))</code>
Name of the property in the datasource that contains open value for the series.	<code>Property:Open@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Open("y").Add();}))</code>	<code>Property:Open@Html.EJS().Chart("container").Series(sr => {sr.Open("y").Add();}))</code>
Option to add trendlines to chart.	<code>Property:TrendLines@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines().Add();}))</code>	<code>Property:TrendLines@Html.EJS().Chart("container").Series(sr => {sr.TrendLines().Add();}))</code>
Options for customizing the appearance of the series or data point while highlighting.	<code>Property:HighlightSettings@(Html.EJ().Chart("chartContainer").Series(sr => {sr.HighlightSettings().Add();}))</code>	Not applicable.

Options for customizing the appearance of the series/data point on selection.	Property: SelectionSettings@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.SelectionSettings().Add();}))	Not applicable.
---	---	-----------------

marker

visibility of marker	Property: Marker.Visible@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Visible(true)).Add();}))	Property: Visible@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.Visible(true)).Add();}))
Fill for marker	Property: Fill@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Fill("red")).Add();}))	Property: Fill@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.Fill("red")).Add();}))
Opacity for marker	Property: Opacity@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Opacity("0.5")).Add();}))	Property: Opacity@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.Opacity("0.5")).Add();}))
Shape of marker	Property: Shape@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Shape("Circle")).Add();}))	Property: Shape@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.Shape("Circle")).Add();}))
Image Url of marker	Property: ImageUrl@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.ImageUrl("")).Add();}))	Property: ImageUrl@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.ImageUrl("")).Add();}))
Border of marker	Property: Border@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Border(b => b.Color("red").Width("2"))).Add();}))	Property: Border@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.IBorder(b => b.Color("red").Width("2"))).Add();}))
Height of marker	Property: Size.Height@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Size(s => s.Height("30"))).Add();}))	Property: Height@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.Height("30"))).Add();}))

Width of marker	Property:Size.Width @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Size(s => s.Width("30"))).Add();}))	Property:Width @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.Width("30"))).Add();})
DataLabelSettings of marker	Property:Marker.DataLabel @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.Size(s => s.DataLabel()).Add();}))	Property:Marker.DataLabel @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel()).Add();})
Visibility of dataLabel	Property:DataLabel.Visible @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Visible(true))).Add();}))	Property:DataLabel.Visible @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Visible(true))).Add();})
Text mapping name of dataLabel	Property:DataLabel.TextMappingName @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.TextMappingName()).Add();}))	Property:DataLabel.Name @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Name()).Add();})
Fill color of data label	Property:DataLabel.Fill @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Fill("red"))).Add();}))	Property:DataLabel.Fill @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Fill("red"))).Add();})
Opacity of data label	Property:DataLabel.Opacity @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Opacity("0.5"))).Add();}))	Property:DataLabel.Opacity @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Opacity("0.5"))).Add();})
Text position of data label	Property:DataLabel.TextPosition @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.TextPosition("middle"))).Add();}))	Property:DataLabel.Position @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.TextPosition("middle"))).Add();})
Alignment of data label	Property:DataLabel.VerticalAlignment @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.VerticalAlignment("near"))).Add();}))	Property:DataLabel.Alignment @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Alignment("near"))).Add();})
Border of data label	Property:DataLabel.Border @(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Border(b => b.Color("blue").Width("2"))).Add();}))	Property:DataLabel.Border @Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Border(b => b.Color("blue").Width("2"))).Add();})

Offset for data label	Property: DataLabel.Offset@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Offset(off => off.X("5").Y("6")))).Add();}))	Not Applicable
Margin of data label	Property: DataLabel.Margin@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Margin(m => m.Top("10").Bottom("10").Left("10").Right("10")).Add();}))	Property: DataLabel.Margin@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Margin(m => m.Top("10").Bottom("10").Left("10").Right("10")).Add();}))
Font of data label	Property: DataLabel.Font@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Font(f => f.FontFamily('SegoeUI').FontStyle('italic').FontWeight("600").Opacity("0.5").Add();}))	Property: DataLabel.Font@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Font(f => f.FontFamily('SegoeUI').FontStyle('italic').FontWeight("600").Opacity("0.5").Add();}))
HTML template in data label	Property: DataLabel.Template@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Template('Chart') .Add();}))	Property: DataLabel.Template@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Template('Chart') .Add();}))
Rounded corner radius X	Not Applicable	Property: dataLabel.Rx@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Rx("5").Add();}))
Rounded corner radius Y	Not Applicable	Property: dataLabel.Ry@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.Ry("10").Add();}))
Maximum Label width for data label	Property: DataLabel.MaximumLabelWidth@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.MaximumLabelWidth("20").Add();}))	Not Applicable
Enable wrapping of text for	Property: DataLabel.EnableWrap@(Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.EnableWrap(true)).Add();}))	Not Applicable

data label		
To show contrast color for data label	Property:DataLabel.ShowContrastColor@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.ShowContrastColor(true)).Add();}))	Not Applicable
To show edge label for data label	Property:DataLabel.ShowEdgeLabels@Html.EJS().Chart("container").Series(sr => {sr.Marker(mar => mar.DataLabel(d => d.ShowEdgeLabels(true)).Add();}))	Not Applicable

TrendLines

Behaviour	API in Essential JS 1	API in Essential JS 2
Trendline settings	Property:Series.TrendLines@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines().Add();}))	Property:Series.TrendLines@Html.EJS().Chart("container").Series(sr => {sr.TrendLines().Add();}))
Visibility of trend line	Property:TrendLines.Visibility@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Visibility('visible')).Add();}))	Not applicable
Type of trend Line	Property:TrendLines.Type@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Type(TrendlineType.Linear)).Add();}))	Property:TrendLines.Type@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Type(Syncfusion.EJ2.Charts.TrendlineTypes.Linear)).Add();}))
Name of trend Line	Property:TrendLines.Name@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Name("TrendLines")).Add();}))	Property:TrendLines.Name@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Name("TrendLines")).Add();}))
Period of	Property:TrendLines.Period@ (Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Period("45")).Add();}))	Property:TrendLines.Period@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Period("45")).Add();}))

trend Line		
Polynomial order for Polynomial type trend Lines	Property:TrendLines.PolynomialOrder@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.PolynomialOrder("3")).Add();}))	Property:TrendLines.PolynomialOrder@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.PolynomialOrder("3")).Add();}))
Backward forecast for trend Lines	Property:TrendLines.Backwardforecast@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Backwardforecast("3")).Add();}))	Property:TrendLines.Backwardforecast@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Backwardforecast("3")).Add();}))
Forward forecast for trend Lines	Property:TrendLines.ForwardForecast@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.ForwardForecast("3")).Add();}))	Property:TrendLines.ForwardForecast@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.ForwardForecast("3")).Add();}))
Fill for trend Lines	Property:TrendLines.Fill@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Fill("red")).Add();}))	Property:TrendLines.Fill@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Fill("red")).Add();}))
Width for trend Lines	Property:TrendLines.Width@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Width("2")).Add();}))	Property:TrendLines.Width@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Width("2")).Add();}))
Intercept value for trend Lines	Property:TrendLines.Intercept@(Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Intercept("2")).Add();}))	Property:TrendLines.Intercept@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Intercept("2")).Add();}))
Legend shape for	Not Applicable	Property:TrendLines.LegendShape@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr =>

trend Lines		<code>tr.LegendShape("Rectangle").Add();</code>
Animation settings for trend Lines	Not Applicable	<code>Property:TrendLines.Animation@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Animation(a => a.Enable(true))).Add();})</code>
Marker settings for trend Lines	Not Applicable	<code>Property:TrendLines.Marker@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.Marker(m => m.Visible(true))).Add();})</code>
Tooltip for trend Lines	<code>Property:TrendLines.Tooltip@Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.Tooltip()).Add();})</code>	<code>Property:TrendLines.EnableTooltip@Html.EJS().Chart("container").Series(sr => {sr.TrendLines(tr => tr.EnableTooltip(true)).Add();})</code>
Dash Array for trend Lines	<code>Property:TrendLines.DashArray@Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.DashArray("10,5")).Add();})</code>	Not Applicable.
Visible on legend for trend Lines	<code>Property:TrendLines.VisibleOnLegend@Html.EJ().Chart("chartContainer").Series(sr => {sr.TrendLines(tr => tr.VisibleOnLegend(true)).Add();})</code>	Not Applicable.

StripLines

Behaviour	API in Essential JS 1	API in Essential JS 2
Default behaviour for striplines	<code>Property:PrimaryXAxis.StripLines@Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.Visible(true)))</code>	<code>Property:PrimaryXAxis.StripLines@Html.EJS().Chart("container").PrimaryXAxis(px => px.StripLines(st => st.Visible(true)))</code>

border for stripline	<code>Property:StripLines.BorderColor@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.BorderColor("pink"))))</code>	<code>Property:stripLines.Border@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.Border(b => b.Color("pink").Width("2"))))</code>
Background color for stripline	<code>Property:StripLines.Color@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.Color("pink"))))</code>	<code>Property:StripLines.Corder@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.Color("pink")))</code>
Start value for stripline	<code>Property:StripLines.Start@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.Start("10"))))</code>	<code>Property:StripLines.Start@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.Start("10")))</code>
End value for stripline	<code>Property:StripLines.End@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.End("20"))))</code>	<code>Property:StripLines.End@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.End("10")))</code>
Start from Axis for stripline	<code>Property:StripLines.StartFromAxis@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.StartFromAxis(true))))</code>	<code>Property:StripLines.StartFromAxis@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.StartFromAxis(true)))</code>
Text in stripline	<code>Property:StripLines.Text@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.Text("text"))))</code>	<code>Property:StripLines.Text@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.Text("text")))</code>
Text alignment in stripline	<code>Property:StripLines.TextAlignment@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.TextAlignment("far"))))</code>	<code>Property:stripLines.HorizontalAlignment@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.HorizontalAlignment("middle")))</code>
Vertical Text alignment in stripline	Not Applicable	<code>Property:StripLines.VerticalAlignment@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.VerticalAlignment("far")))</code>

Size of stripline	<code>Property:StripLines.Width@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.Width("10"))))</code>	<code>Property:StripLines.Size@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.Size("10")))</code>
ZIndex of stripline	<code>Property:StripLines.ZIndex@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.ZIndex("Behind"))))</code>	<code>Property:stripLines.size@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.ZIndex("Behind")))</code>
Font style of stripline	<code>Property:StripLines.FontStyle@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.StripLines(st => st.FontStyle())))</code>	<code>Property:StripLines.TextStyle@Html.EJS().Chart("container")PrimaryXAxis(px => px.StripLines(st => st.FontStyle()))</code>

Multilevel Labels

Behaviour	API in Essential JS 1	API in Essential JS 2
Default behaviour for multilevel labels	<code>Property:PrimaryXAxis.MultilevelLabels@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels()))</code>	<code>Property:PrimaryXAxis.MultilevelLabels@Html.EJS().Chart("container")PrimaryXAxis(px => px.MultilevelLabels())</code>
Text alignment for multilevel labels	<code>Property:MultiLevelLabels.TextAlignment@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.TextAlignment("Near")))</code>	<code>Property:MultilevelLabels.Alignment@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.Alignment("Near")))</code>
Text overflow for multilevel labels	<code>Property:multiLevelLabels.TextOverFlow@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.TextOverFlow("Trim")))</code>	<code>Property:MultiLevelLabels.OverFlow@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.OverFlow("Trim")))</code>
Border for multilevel labels	<code>Property:multiLevelLabels.Border@(Html.EJ().Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.Border(b => b.Width("2").Color("red"))))</code>	<code>Property:MultiLevelLabels.Border@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.Border(b => b.Width("2").Color("red")))</code>

Start value for label	Property:MultiLevelLabels.Start@ (Html.EJ() .Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.Start("45"))))	Property:MultiLevelLabels.Categories.Start@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.Categories(c => c.Start("45"))))
End value for label	Property:MultiLevelLabels.End@ (Html.EJ() .Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.End("50"))))	Property:MultiLevelLabels.Categories.End@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.Categories(c => c.End("50"))))
Text for label	Property:MultiLevelLabels.Text@ (Html.EJ() .Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.Text("start"))))	Property:MultiLevelLabels.Categories.Text@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.Categories(c => c.Text("start"))))
maximum text width for label	Property:MultiLevelLabels.MaximumTextWidth@ (Html.EJ() .Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.MaximumTextWidth("20"))))	Property:MultiLevelLabels.Categories.MaximumTextWidth@Html.EJ().Chart("container").PrimaryXAxis(px => px.MultilevelLabels(m => m.Categories(c => c.MaximumTextWidth("20"))))
level of labels	Property:MultiLevelLabels.Level@ (Html.EJ() .Chart("chartContainer").PrimaryXAxis(px => px.MultilevelLabels(m => m.Level("2"))))	Not applicable

Methods

Behavior	API in Essential JS 1	API in Essential JS 2
animation for series	Property:Animate@ (Html.EJ() .Chart("chartContainer").Animate())	Not applicable
Redraw for chart	Property:Redraw@ (Html.EJ() .Chart("chartContainer").Redraw())	Property:Refresh@Html.EJ().Chart("container").Refresh()
Export	Property:Export()@ (Html.EJ() .Chart("chartContainer").Export())	Property:Export()@Html.EJ().Chart("container").Export()
Print	Property:Print()@ (Html.EJ() .Chart("chartContainer").Print())	Property:Print()@Html.EJ().Chart("container").Print()
AddSeries	Not Applicable	Property:AddSeries()@Html.EJ().Chart("container").AddSeries()
RemoveSeries	Not Applicable	Property:RemoveSeries()@Html.EJ().Chart("container").RemoveSeries()

Events

Behaviour	API in Essential JS 1	API in Essential JS 2
Fires on annotation click	<code>Property:AnnotationClick@(Html.EJ().Chart("chartContainer").AnnotationClick())</code>	Not applicable
Fires after animation	<code>Property:AnimationComplete@(Html.EJ().Chart("chartContainer").AnimationComplete())</code>	<code>Property:AnimationComplete()@Html.EJ().Chart("container").AnimationComplete()</code>
Fires on axis label click	<code>Property:AxisLabelClick@(Html.EJ().Chart("chartContainer").AxisLabelClick())</code>	Not applicable
Fires before axis label render	<code>Property:AxisLabelRendering@(Html.EJ().Chart("chartContainer").AxisLabelRendering())</code>	<code>Property:AxisLabelRender()@Html.EJ().Chart("container").AxisLabelRender()</code>
Fires on axis label mouse move	<code>Property:AxisLabelMouseMove@(Html.EJ().Chart("chartContainer").AxisLabelMouseMove())</code>	Not applicable
Fires on axis label initialize	<code>Property:AxisLabelInitialize@(Html.EJ().Chart("chartContainer").AxisLabelInitialize())</code>	Not applicable
Fires before axis range	<code>Property:AxesRangeCalculate@(Html.EJ().Chart("chartContainer").AxesRangeCalculate());</code>	<code>Property:AxisRangeCalculated()@Html.EJ().Chart("container").AxisRangeCalculated()</code>

e calcu latio n		
Fires on axis title rend ering	Property:AxisTitleRendering@(Html.EJ().Chart("chartContainer").AxisTitleRendering())	Not applicable
Fires on after chart resiz e	Property:AfterResize@(Html.EJ().Chart("chartContainer").AfterResize())	Not applicable
Fires on befor e chart resiz e	Property:BeforeResize@(Html.EJ().Chart("chartContainer").BeforeResize())	Property:Resized@Html.EJ().Chart("container").Resized()
Fires on chart click	Property:ChartClick@(Html.EJ().Chart("chartContainer").ChartClick())	Property:ChartMouseClicked@Html.EJ().Chart("container").ChartMouseClicked()
Fires on chart mou se mov e	Property:ChartMouseMove@(Html.EJ().Chart("chartContainer").ChartMouseMove())	Property:ChartMouseMove@Html.EJ().Chart("container").ChartMouseMove()
Fires on chart mou se leave	Property:ChartMouseLeave@(Html.EJ().Chart("chartContainer").ChartMouseLeave())	Property:ChartMouseLeave@Html.EJ().Chart("container").ChartMouseLeave()
Fires on befor	Property:ChartDoubleClick@(Html.EJ().Chart("chartContainer").ChartDoubleClick())	Not applicable

e chart doub le click		
Fires on chart mouse up	Not Applicable	Property:ChartmouseUp@Html.EJ().Chart("container").ChartmouseUp()
Fires on chart mouse down	Not Applicable	Property:ChartmouseDown@Html.EJ().Chart("container").ChartmouseDown()
Fires durin g the calcu lation of chart area bound ds.You can use this even t to custo mize the bound ds of chart area	Property:ChartAreaBoundsCalculate@(Html.EJ().Chart("chartContainer").ChartAreaBoundsCalculate())	Not applicable
Fires when the drag ging is	Property:DragStart@(Html.EJ().Chart("chartContainer").DragStart())	Not applicable

start ed		
Fires while drag ging	<code>Property:Dragging@ (Html.EJ() .Chart ("chartCo ntainer") .Dragging())</code>	Not applicable
Fires whe n the drag ging is com plete d	<code>Property:DragEnd@ (Html.EJ() .Chart ("chartCo ntainer") .DragEnd())</code>	<code>Property:DragComplete@Html.EJ() .Char t ("container") .DragComplete())</code>
Fires whe n chart is destr oyed com plete ly.	<code>Property:Destroy@ (Html.EJ() .Chart ("chartCo ntainer") .Destroy())</code>	Not applicable
Fires after chart is creat ed.	<code>Property:Create@ (Html.EJ() .Chart ("chartCon tainer") .Create())</code>	<code>Property:Loaded@Html.EJ() .Chart ("co ntainer") .Loaded())</code>
Fires befor e rend ering the data label s.	<code>Property:DisplayTextRendering@ (Html.EJ() .Char t ("chartContainer") .DisplayTextRenderin g())</code>	<code>Property:TextRender@Html.EJ() .Chart (" container") .TextRender())</code>
Fires, whe n error	<code>Property:ErrorBarRendering@ (Html.EJ() .Chart (" chartContainer") .ErrorBarRendering())</code>	Not applicable

bar is rendering.		
Fires during the calculation of legend bounds.	Property:LegendBoundsCalculate@(Html.EJ().Chart("chartContainer").LegendBoundsCalculate())	Not applicable
Fires on clicking the legend item.	Property:LegendItemClick@(Html.EJ().Chart("chartContainer").LegendItemClick())	Not applicable
Fires when moving mouse over legend item	Property:LegendItemMouseMove@(Html.EJ().Chart("chartContainer").LegendItemMouseMove())	Not applicable
Fires before rendering the legend item.	Property:LegendItemRendering@(Html.EJ().Chart("chartContainer").LegendItemRendering())	Property:LegendRender@Html.EJ().Chart("container").LegendRender()
Fires before	Property:Load@(Html.EJ().Chart("chartContainer").Load())	Property:Load@Html.EJ().Chart("container").Load()

e loadi ng the chart .		
Fires, whe n multi level label s are rend ering .	Property:MultiLevelLabelRendering@ (Html.EJ().Chart("chartContainer").MultiLevelLabelRendering())	Property:AxisMultiLabelRender@Html.EJ().Chart("container").AxisMultiLabelRender()
Fires on clicki ng a point in chart .	Property:PointRegionClick@ (Html.EJ().Chart("chartContainer").PointRegionClick())	Property:PointClick @Html.EJ().Chart("container").PointClick()
Fires whe n mou se is mov ed over a point .	Property:PointRegionMouseMove@ (Html.EJ().Chart("chartContainer").PointRegionMouseMove())	Property:PointMove @Html.EJ().Chart("container").PointMove()
Fires befor e rend ering chart .	Property:PreRender@ (Html.EJ().Chart("chartContainer").PreRender())	Not applicable

Fires when point render.	Not Applicable	Property:PointRender @Html.EJ().Chart("container").PointRender()
Fires after selected the data in chart.	Property:RangeSelected@ (Html.EJ().Chart("chartContainer").RangeSelected())	Not applicable
Fires after selecting a series.	Property:SeriesRegionClick@ (Html.EJ().Chart("chartContainer").SeriesRegionClick())	Not applicable
Fires before rendering a series.	Property:SeriesRendering@ (Html.EJ().Chart("chartContainer").SeriesRendering())	Property:SeriesRender @Html.EJ().Chart("container").SeriesRender()
Fires before rendering the marker symbols.	Property:SymbolRendering@ (Html.EJ().Chart("chartContainer").SymbolRendering())	Not applicable
Fires before rendering	Property:TrendlineRendering@ (Html.EJ().Chart("chartContainer").TrendlineRendering())	Not applicable

ering the trend line		
Fires before rendering the Chart title.	<code>Property:TitleRendering@(Html.EJ().Chart("chartContainer").TitleRendering())</code>	Not applicable
Fires before rendering the Chart sub title.	<code>Property:SubTitleRendering@(Html.EJ().Chart("chartContainer").SubTitleRendering())</code>	Not applicable
Fires before rendering the tooltip.	<code>Property:ToolTipInitialize@(Html.EJ().Chart("chartContainer").ToolTipInitialize())</code>	<code>Property:ToolTipRender @Html.EJ().Chart("container").ToolTipRender()</code>
Fires before rendering cross hair tooltip in axis	<code>Property:TrackAxisToolTip@(Html.EJ().Chart("chartContainer").TrackAxisToolTip())</code>	Not applicable
Fires before rendering	<code>Property:TrackToolTip@(Html.EJ().Chart("chartContainer").TrackToolTip())</code>	Not applicable

track ball toolti p.		
Even t trigg ered whe n scroll start s.	<code>Property:ScrollStart@(Html.EJ().Chart("chartContainer").ScrollStart())</code>	<code>Property:ScrollStart @Html.EJ().Chart("container").Scr ollStart()</code>
Even t trigg ered whe n scroll ends.	<code>Property:ScrollEnd@(Html.EJ().Chart("chartContainer").ScrollEnd())</code>	<code>Property:ScrollEnd@Html.EJ().Chart("c ontainer").ScrollEnd()</code>
Even t trigg ered whe n scroll chan ges.	<code>Property:ScrollChange@(Html.EJ().Chart("chartContainer").ScrollChange())</code>	<code>Property:ScrollChange@Html.EJ().Chart("co ntainer").ScrollChange()</code>
Fires while perfo rmin g recta ngle zoo ming in chart .	<code>Property:ZoomComplete@(Html.EJ().Chart("chartContainer").ZoomComplete())</code>	<code>Property:ZoomComplete@Html.EJ().Cha rt("container").ZoomComplete()</code>

Chart properties

Behaviour	API in Essential JS 1	API in Essential JS 2
selected data index	Property:SelectedDataPointIndexes:@ (Html.EJ() .Chart ("chartContainer") .SelectedDataPointIndexes (s => s.SeriesIndex ("0") .PointIndex ("1"))))	Property:SelectedDataIndexes@Html.EJ() .Chart ("container") .SelectedDataPointIndexes (s => s.SeriesIndex ("0") .PointIndex ("1"))
sideBySide Series Placement for column based series	Property:SideBySideSeriesPlacement:@ (Html.EJ() .Chart ("chartContainer") .SideBySideSeriesPlacement (true))	Property:SideBySidePlacement@Html.EJ() .Chart ("container") .SideBySidePlacement (true)
ZoomSettings	Property:Zooming:@ (Html.EJ() .Chart ("chartContainer") .Zooming (z => z.Enable (true) .EnablePinch (true) .EnableScrollBar (true)))	Property:ZoomSettings@Html.EJ() .Chart ("container") .ZoomSettings (z => z.Enable (true) .EnablePinch (true) .EnableScrollBar (true)))
Background color of the chart	Property:Background @ (Html.EJ() .Chart ("chartContainer") .Background ('transparent'))	Property:Background@Html.EJ() .Chart ("container") .Background ('transparent')
URL of the image to be used as chart background.	Property:BackGroundImageUrl @ (Html.EJ() .Chart ("chartContainer") .BackGroundImageUrl (""))	Not Applicable
Customizing border of the chart	Property:Border @ (Html.EJ() .Chart ("chartContainer") .Border (b => b.Color ("red") .Width ("2"))))	Property:Border@Html.EJ() .Chart ("container") .Border (b => b.Color ("red") .Width ("2"))))
This provides options for customizing export settings	Property:ExportSettings@ (Html.EJ() .Chart ("chartContainer") .ExportSettings (e => e.FileName ("chart") .Angle ("45"))))	Property:Export()@Html.EJ() .Chart ("container") .Export (e => e.FileName ("chart") .Angle ("45"))))
ChartArea customization	Property:ChartArea@ (Html.EJ() .Chart ("chartContainer") .ChartArea (c => c.Background ("transparent") .Border (b => b.Opacity ("0.3") .Color ("red") .Width ("2"))))	Property:ChartArea@Html.EJ() .Chart ("container") .ChartArea (c => c.Background ("transparent") .Border (b => b.Opacity ("0.3") .Color ("red") .Width ("2"))))

ASP.NET MVC Scaffolding

Syncfusion includes an extension for **Visual Studio Scaffolding** for the Syncfusion ASP.NET MVC platform to quickly add code that interacts with data models and reduce the amount of time to develop with data operation in your project. Scaffolding provides an easier way to create Views and Controller action methods for Syncfusion ASP.NET MVC Chart controls.

Note: The Syncfusion ASP.NET MVC UI Scaffold is available from **v16.4.0.40**.

Getting Started

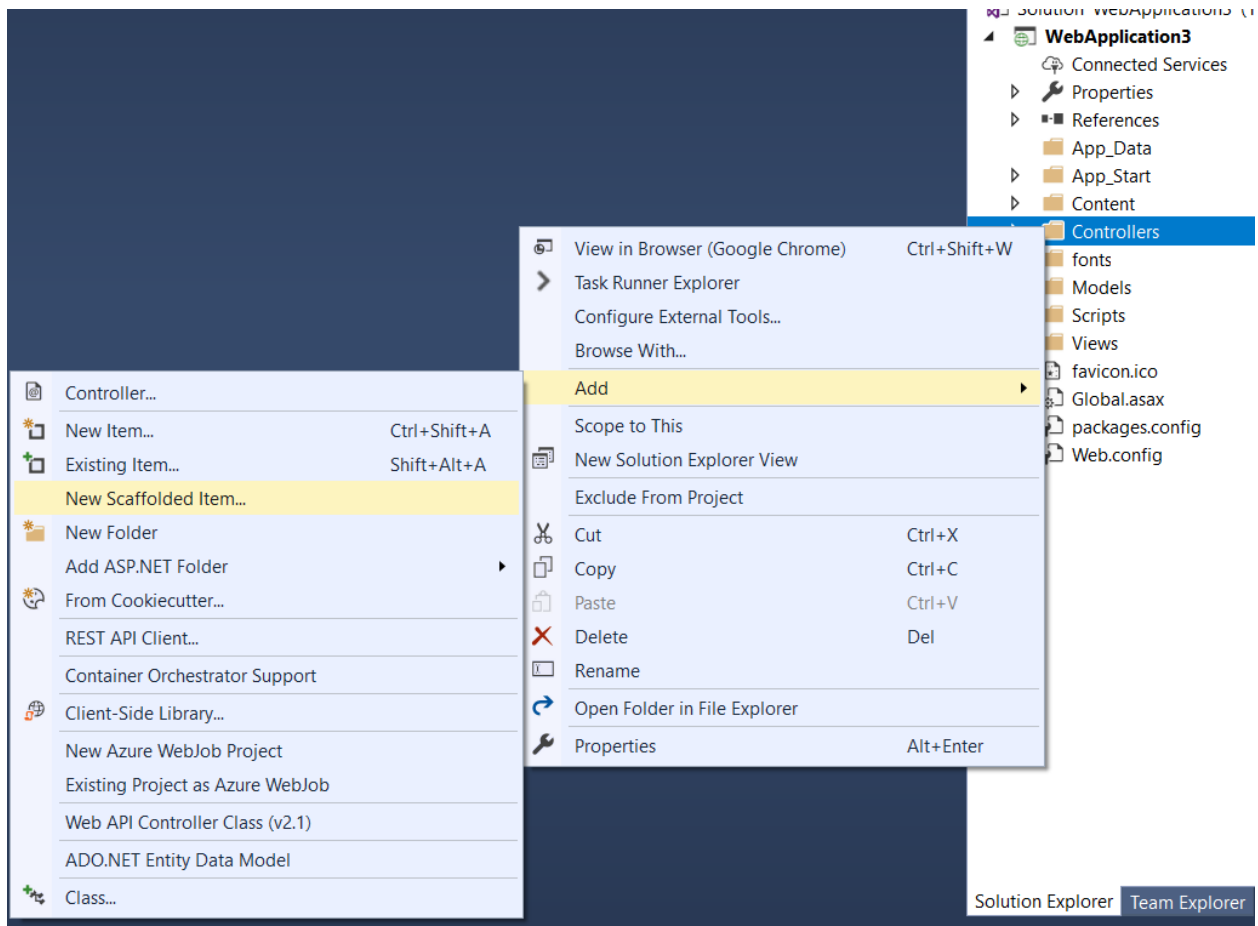
Let's start with the steps on how to scaffold the ASP.NET MVC Chart into your web application.

- Create an ASP.NET MVC application and add an Entity Framework data model referring from the [documentation](#) with Scheduler related fields such as Id, Subject, Location, Start Date, End Date and All-day. Once the model file is added, ensure the required DbContext and all its related properties are added.
- Refer the [Getting Started documentation](#) to know about how to configure the Syncfusion Essential JS2 for ASP.NET MVC in your web application.

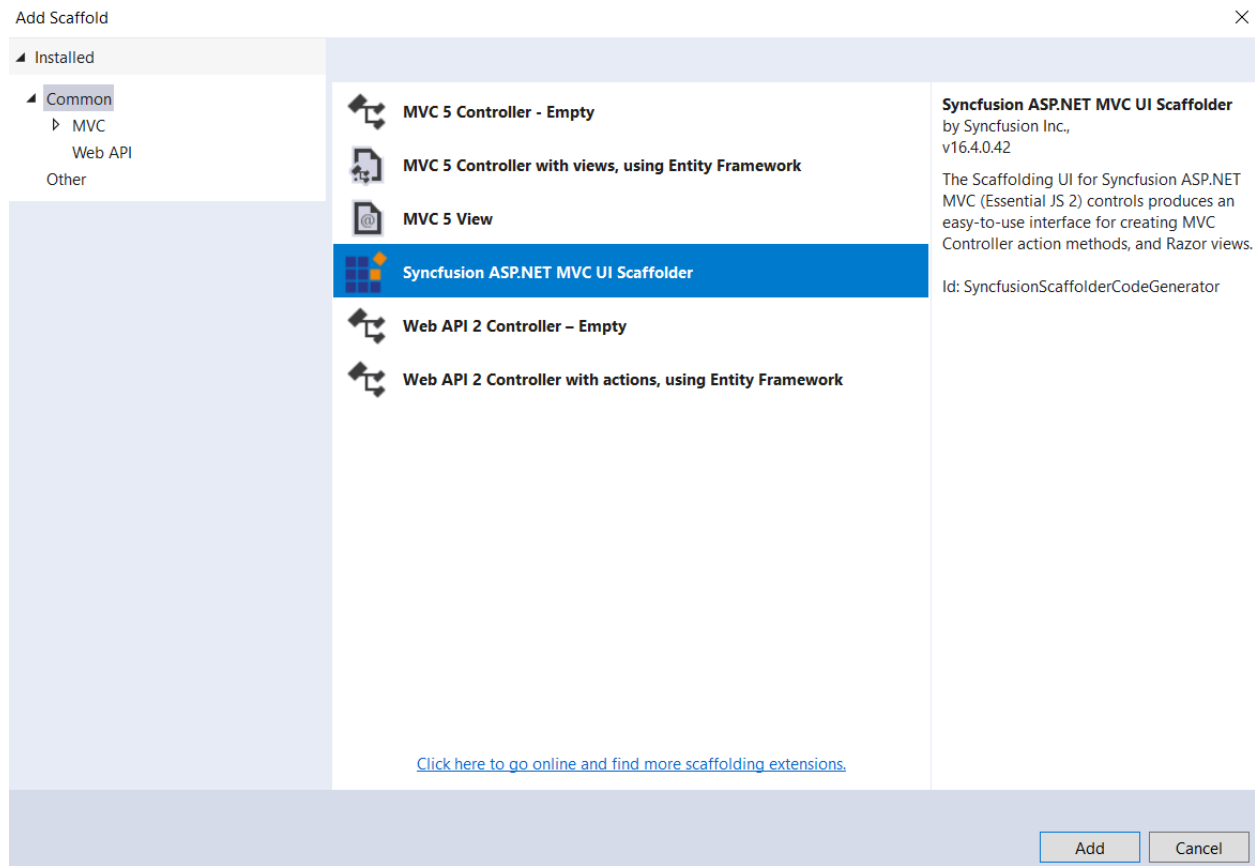
Add a Scaffolded Item

The following steps direct you to add a Scaffolded item to your ASP.NET MVC Web Application.

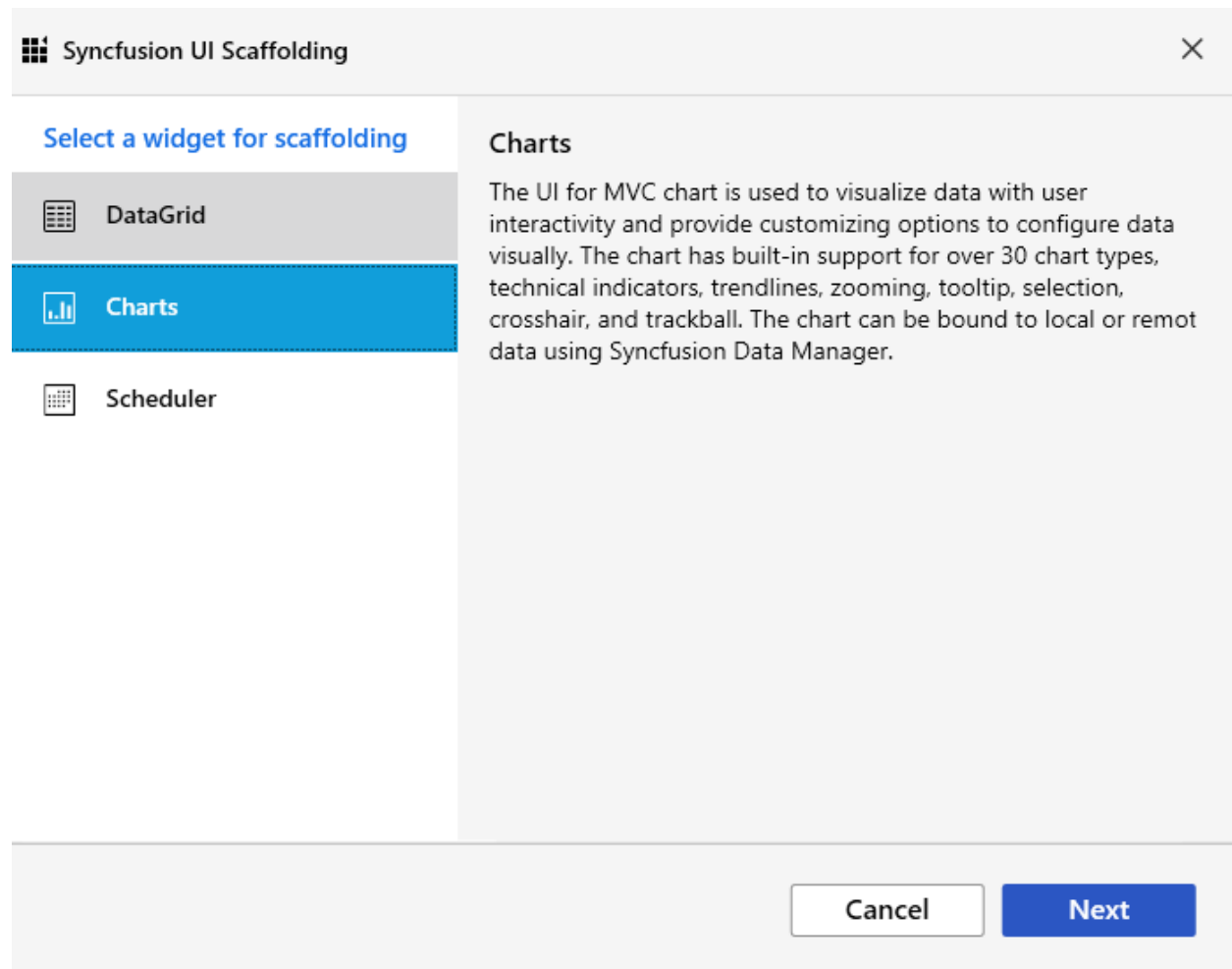
- Right-click on Controller folder in the Solution Explorer and select **Add → New Scaffolded Item**.




- You will see the Add Scaffold dialog. Select **Syncfusion ASP.NET MVC UI Scaffolder** and click 'Add' button, which will display the Syncfusion UI Scaffolding dialog.



- Select the Chart control to perform Scaffolding and click **Next**.



- Syncfusion UI Scaffolding for the selected control dialog will be opened. Enter the **Controller and View** names as per the application requirement. Also, select the required **Model Class** of the active project and its relevant **Data Context Class** and then Click **Next**.

 Syncfusion UI Scaffolding for Charts ✕

Select the model

Controller Name*:

View Name*:


Model Class*:

Data Context Class*:

Cancel

Next

- Syncfusion UI Scaffolding for the Chart control feature dialog will be opened. Select the required features and Click **Add**. Use the **back arrow** if you want to modify the already provided Controller or View name, and to change the **selected Model Class and Data Context Class**.

 Syncfusion UI Scaffolding for Charts ✕

Select the feature(s)

☐ Tooltip☐ Crosshair☐ Zooming


Series Type

Line

Series Option*:

X-axis

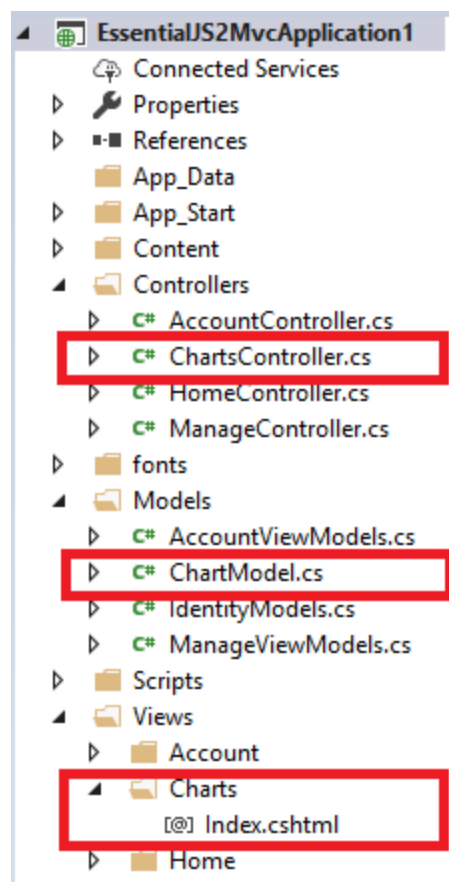
Y-axis



CancelAdd

select the corresponding Chart **X-axis** and **Y-axis** from option.

- The **Controller** and the corresponding **View** files are now generated with the selected features code snippet.



Note: Ensure that at least one Entity Framework model exists in your project and the application gets compiled once. If no Entity Framework model exists in your application, refer to the [documentation](#) to generate the Entity Framework model. Once model file added, ensure required DbContext and properties are added. Now, build the application and try scaffolding. If any changes done in model properties later, compile the application once before perform scaffolding.

How to render Syncfusion Control?'

Refer to the following UG links to render Syncfusion control after performed scaffolding.

MVC4: [Configure Essential JS 2 using Syncfusion.EJ2.MVC4 package](#)

MVC5: [Configure Essential JS 2 using Syncfusion.EJ2.MVC5 package](#)

How To

<!-- markdownlint-disable MD036 -->

Ajax Chart in ASP.NET MVC Chart control

The following section explains how to customize various aspects of the chart.

Bind data to chart on ajax request

You can bind data to chart on ajax request using the following steps:

Step 1:

Create a http post action method to return data for ajax call in a controller page. The following GetServerData() method will return a JSON data string.

```
`cs
[HttpPost]
public ActionResult GetServerData()
{
    List<LineChart> data1 = new List<LineChart>();
    data1.Add(new LineChart("India", 1));
    data1.Add(new LineChart("USA", 5));
    data1.Add(new LineChart("USSR", 10));
    data1.Add(new LineChart("Chinaa", 12));
    data1.Add(new LineChart("Japan", 8));
    return Json(data1);
}

public class LineChart
{
    public LineChart(string name, double cnt)
    {
        this.x = name;
        this.y = cnt;
    }
    public string x { get; set; }
    public double y { get; set; }
}
`
```

Step 2:

Make a simple chart with series.

```
`html
@using Syncfusion.EJ2;
@using EJ2MVCSamples.Controllers;
<div class="col-md-8">
    @Html.EJS().Chart("container").PrimaryXAxis(px =>
        px.Title("Years").ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Series(series =>
        {
            series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Add();
        }
    )
</div>
```

```
}  
).Render()  
</div>  
,
```

Step 3:

Make ajax call to the `GetServerData()` action method in chart load event to get JSON data, and then assign the data to the chart series using the `dataSource` property and mapping the field name to `xName` and `yName` properties of the series.

```
`html  
@using Syncfusion.EJ2;  
@using EJ2MVCSamples.Controllers;  
<div class="col-md-8">  
@Html.EJS().Chart("container").PrimaryXAxis(px =>  
px.Title("Years").ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Series(series =>  
{  
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Add();  
}  
).Load("chartLoad").Render()  
</div>  
<script>  
var chartLoad = function (args) {  
$.ajax({  
type: "POST",  
url: '@Url.Action("GetServerData","Home")',  
async: false,  
success: function (data) {  
args.chart.series[0].dataSource = data;  
args.chart.series[0].xName = "x";  
args.chart.series[0].yName = "y";  
}  
});  
}  
</script>  
,
```

CSHTML

```

    @Html.EJS().Chart("container").PrimaryXAxis(px =>
px.Title("Years").ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Series
s(series =>
    {

series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).Name("Product
X").Add();
    }
).Load("chartLoad").Render()
<script>
    var chartLoad = function (args) {
        $.ajax({
            type: "POST",
            url: '@Url.Action("GetServerData","ajaxdata")',
            async: false,
            success: function (data) {
                args.chart.series[0].dataSource = data;
                args.chart.series[0].xName = "x";
                args.chart.series[0].yName = "y";
            }
        });
    }
</script>

```

AJAX-DATA.CS

```

public ActionResult Index()
{
    return View();
}
[HttpPost]
public ActionResult GetServerData()
{
    List<LineChart> data1 = new List<LineChart>();
    data1.Add(new LineChart("India", 1));
    data1.Add(new LineChart("USA", 5));
    data1.Add(new LineChart("USSR", 10));
    data1.Add(new LineChart("Chinaa", 12));
    data1.Add(new LineChart("Japan", 8));
    return Json(data1);
}
public class LineChart
{
    public LineChart(string name, double cnt)
    {
        this.x = name;
        this.y = cnt;
    }
    public string x { get; set; }
    public double y { get; set; }
}

```

Sample reference

Sample for how to get data from ajax call is available in [chart sample](#).

```
<!-- markdownlint-disable MD036 -->
```

Customize the background color of data label template from webservice

Bind the text and interior information for a point from dataSource. To change the background color in the datalabel template, use `${point.text}`. You have to bind the property from dataSource to name in the data label options to use point.text.

Follow the given steps to customize the background color of data label template from webservice, which includes colors:

Step 1: Get the data from webservice, which also includes color attribute to bind the name in the data label options to use point.text in the data label template.

To get data from web service, use any adaptor webApi or Url adaptor in the data manager, and then bind the data manager to the `dataSource` property in the series options.

```
`html
@Html.EJS().Chart("container").Series(
series =>
{
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).
XName("Year").YName("yValue").DataSource("dataManager").
Query("query").Add();
}).PrimaryXAxis(
px => px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
).Render()
<script>
var dataManager = new ej.data.DataManager({
url: 'http://localhost:54290/api/Values',
adaptor: new ej.data.WebApiAdaptor
});
var query = new ej.data.Query();
</script>
`cs
// get the data from here
public object Get()
{
```

```

List<EmployeeModel> revenue = new List<EmployeeModel>();
revenue.Add(new EmployeeModel() { Year = "2018-06-13T00:00:00", yValue = 30, color = "red" });
revenue.Add(new EmployeeModel() { Year = "2018-06-14T00:00:00", yValue = 20, color = "blue" });
revenue.Add(new EmployeeModel() { Year = "2018-06-15T00:00:00", yValue = 15, color = "green" });
revenue.Add(new EmployeeModel() { Year = "2018-06-16T00:00:00", yValue = 30, color = "orange" });
revenue.Add(new EmployeeModel() { Year = "2018-06-17T00:00:00", yValue = 60, color = "skyblue" });
revenue.Add(new EmployeeModel() { Year = "2018-06-18T00:00:00", yValue = 10, color = "red" });
revenue.Add(new EmployeeModel() { Year = "2018-06-19T00:00:00", yValue = 70, color = "yellow" });
revenue.Add(new EmployeeModel() { Year = "2018-06-20T00:00:00", yValue = 50, color = "pink" });
revenue.Add(new EmployeeModel() { Year = "2018-06-21T00:00:00", yValue = 88, color = "green" });
revenue.Add(new EmployeeModel() { Year = "2018-06-22T00:00:00", yValue = 35, color = "blue" });
return new { Items = revenue, Count = revenue.Count() };
}

public class EmployeeModel
{
    public string Year { get; set; }
    public double yValue { get; set; }
    public string color { get; set; }
}

```

Step 2: Initialize the data label template <div> as demonstrated in the following html page.

```

`html
<script id="index" type="text/x-template">
<div id='templateWrap' style="background-color: ${point.text}; border-radius: 3px;">

<div class='des' style="color:black; font-family:Roboto; font-style: normal; font-size:16px;padding-
right:6px">
<span style="color:white;">${point.y}</span>
</div>
</div>
</script>

```

Step 3: To show the data label template, set the element id to the `template` property in data label.


```

`html
@Html.EJS().Chart("container").Series(
series =>
{
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).
XName("Year").YName("yValue").DataSource("dataManager").
Query("query").Marker(mark => mark.Visible(true).DataLabel(dl => dl.Visible(true).
Position(Syncfusion.EJ2.Charts.LabelPosition.Top).Name("color").Template("#index"))).Add();
}).PrimaryXAxis(
px => px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
).PrimaryYAxis(
py => py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).
Interval(20).Minimum(0).Maximum(100)
).Render()
`

```

CSHTML

```

@Html.EJS().Chart("container").Series(
    series =>
    {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).
        XName("Year").YName("yValue").DataSource("dataManager").
        Query("query").Marker(mark => mark.Visible(true).DataLabel(dl
=> dl.Visible(true).
Position(Syncfusion.EJ2.Charts.LabelPosition.Top).Name("color").Template("#i
ndex"))).Add();
    }).PrimaryXAxis(
        px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).
EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    ).PrimaryYAxis(
        py =>
py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).
        Interval(20).Minimum(0).Maximum(100)
    ).Render()
<script id="index" type="text/x-template">
    <div id='templateWrap' style="background-color: ${point.text}; border-
radius: 3px;">
        

```

```

        <div class='des' style="color:black; font-family:Roboto; font-style:
normal; font-size:16px;padding-right:6px">
            <span style="color:white;">${point.y}</span>
        </div>
    </div>
</script>
<script>
    var dataManager = new ej.data.DataManager({
        url: 'http://localhost:54290/api/Values',
        adaptor: new ej.data.WebApiAdaptor
    });
    var query = new ej.data.Query();
</script>

```

TEMPLATE.CS

```

public ActionResult Index()
{
    return View();
}

```

Sample: Customize the background color of data label is available in [chart sample](#).

<!-- markdownlint-disable MD036 -->

Render chart in partial view in ajax call

You can render the charts in partial view in ajax call by calling the partial view action method in ajax.

Follow the given steps to render the charts in partial view.

Step 1: Initially, render the home view page, and then call the partial view action method via ajax call in the home view page.

```

`html
<div id="chartContainer">
</div>
<script>
$.ajax({
url: '@Url.Action("Create")',
type: 'POST',
dataType: 'html',
global: false,
success: function (response) {
$('#chartContainer').html(response);
}
});
</script>

```

```
,  
  
`cs  
// partial view action method  
[HttpPost]  
public ActionResult Create()  
{  
    List<ChartData> chartData = new List<ChartData>  
    {  
        new ChartData { xValue = 10, yValue = 21 },  
        new ChartData { xValue = 20, yValue = 24 },  
        new ChartData { xValue = 30, yValue = 36 },  
        new ChartData { xValue = 40, yValue = 38 },  
        new ChartData { xValue = 50, yValue = 54 },  
        new ChartData { xValue = 60, yValue = 57 },  
        new ChartData { xValue = 70, yValue = 70 },  
    };  
    ViewBag.dataSource = chartData;  
    return PartialView("_RefreshCharts");  
}  
  
public class ChartData  
{  
    public double xValue;  
    public double yValue;  
}  
,
```

Step 2: Return the partial view page when calling partial view action method. In that partial view, refer the `scriptManager` to render the charts.

```
`html  
<div>  
    @Html.EJS().Chart("container").Series(sr =>  
    {  
        sr.XName("xValue").YName("yValue").Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).DataSource(  
            ViewBag.dataSource).Add();  
    }).Render();  
</div>
```

```
</div>
```

```
@Html.EJS().ScriptManager()
```

```
,
```

Sample reference: Sample for how to render the charts in partial view is available in [chart sample](#).

To add dotted line using annotation

By using **annotation**, you can add dotted lines in the chart.

To add dotted lines in the chart, follow the given steps:

Step 1: Initialize the custom elements by using the **annotation** property.

By setting **coordinateUnits** value as **point** in annotation object, you can place dotted lines in the chart based on point x and y values.

CSHTML

```
@Html.EJS().Chart("container").ChartArea(area => area.Border(br =>
br.Color("transparent"))).Series(series =>
{
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue")
    .Marker(mr => mr.Visible(true).Width(10).Height(10)).YName("yValue")
    .DataSource(ViewBag.dataSource).Name("Germany").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).Annotations(an =>
{
an.Region(Syncfusion.EJ2.Charts.Regions.Chart).X("2014").Y("50").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).
Content("#templateWrap").Add();
}).Render()
<script id="templateWrap" type="text/x-template">
<div style="border-top:3px dashed grey;border-top-width: 2px; width:
10000px"></div>
</script>
```

DOTTED-ANNOTATION.CS

```
public IActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = "2014", yValue = 21 },
        new LineChartData { xValue = "2015", yValue = 24 },
        new LineChartData { xValue = "2016", yValue = 36 },
        new LineChartData { xValue = "2017", yValue = 38 },
        new LineChartData { xValue = "2018", yValue = 54 },
        new LineChartData { xValue = "2019", yValue = 57 },
        new LineChartData { xValue = "2020", yValue = 70 },
    };
    ViewBag.dataSource = chartData;
```

```

        return View();
    }
    public class LineChartData
    {
        public string xValue;
        public double yValue;
        public double yValue1;
    }

```

<!-- markdownlint-disable MD036 -->

Hide the tooltip for unselected series

By using the [tooltipRender](#) event, you can cancel the tooltip for unselected series in the chart.

To hide the tooltip value in unselected series, follow the given steps:

Step 1: By using the [tooltipRender](#) event, you can get the series elements in the arguments. By using this argument, it can be compared whether seriesElementclasslist is deselected container or not. If it is true then the tooltip can be cancelled by setting the value for `args.cancel` as true.

CSHTML

```

@Html.EJS().Chart("container").ChartArea(area => area.Border(br =>
br.Color("transparent"))).Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).Width(2).XName("xValue")
        .Marker(mr =>
mr.Visible(true).Width(10).Height(10)).YName("yValue")
        .DataSource(ViewBag.dataSource).Name("Germany").Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
        .Width(2).XName("xValue").YName("yValue1")
        .Marker(mr => mr.Visible(true).Width(10).Height(10))
        .DataSource(ViewBag.dataSource).Name("England").Add();
}).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .MajorGridLines(mg => mg.Width(0))
    .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)

.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).LabelFormat("y")
    .PrimaryYAxis(py => py.LabelFormat("{value}%")
    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    .MajorTickLines(mt => mt.Width(0))
    .MinorTickLines(mt => mt.Width(0))
    .LineStyle(ls =>
ls.Width(0)).Interval(20).Minimum(0).Maximum(100)
    ).Title("Inflation - Consumer Price").Tooltip(tt =>
tt.Enable(true)).TooltipRender("tooltipRender").SelectionMode(Syncfusion.EJ2.Charts.SelectionMode.Series).Render()
<script>
    var tooltipRender = function (args) {
        var series = (args.series);
        if (series.seriesElement.classList[0] ===
'container ej2 deselected') {

```

```

        args.cancel = true;
    }
}
</script>

```

HIDE-TOOLTIP.CS

```

public IActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 21, yValue1 = 28 },
        new LineChartData { xValue = new DateTime(2006, 01, 01),
yValue = 24, yValue1 = 44 },
        new LineChartData { xValue = new DateTime(2007, 01, 01),
yValue = 36, yValue1 = 48 },
        new LineChartData { xValue = new DateTime(2008, 01, 01),
yValue = 38, yValue1 = 50 },
        new LineChartData { xValue = new DateTime(2009, 01, 01),
yValue = 54, yValue1 = 66 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 57, yValue1 = 78 },
        new LineChartData { xValue = new DateTime(2011, 01, 01),
yValue = 70, yValue1 = 84 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class LineChartData
{
    public DateTime xValue;
    public double yValue;
    public double yValue1;
}

```

<!-- markdownlint-disable MD036 -->

Create a control chart

You can create a control chart (with UCL, CL, and LCL limits) using the stripline and annotation features.

To create a control charts, follow the given steps.

Step 1: Load line charts for the data.

CSHTML

```

@(Html.EJS().Chart("container").Title("Sales History of Product
X").Load("load").Loaded("loaded").PointRender("pointRender").Series(series
=>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
        XName("xValue").
        YName("yValue").
        Marker(mr => mr.Visible(true).Height(10).Width(10)).
        DataSource(ViewBag.dataSource).Add();
}
)

```

```

    }).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(mg =>
mg.Width(0)).MajorTickLines(mt => mt.Width(0)))
    .PrimaryYAxis(py => py.MajorGridLines(mg =>
mg.Width(0)).MajorTickLines(mt => mt.Width(0)))
    .Render()
)

```

STEP1.CS

```

public IActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(1975, 01, 01),
yValue = 36},
        new LineChartData { xValue = new DateTime(1980, 01, 01),
yValue = 32.5 },
        new LineChartData { xValue = new DateTime(1985, 01, 01),
yValue = 39 },
        new LineChartData { xValue = new DateTime(1990, 01, 01),
yValue = 34.4 },
        new LineChartData { xValue = new DateTime(1995, 01, 01),
yValue = 31.5 },
        new LineChartData { xValue = new DateTime(2000, 01, 01),
yValue = 40 },
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 60 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 66 },
    };
    ViewBag.dataSource = chartData;
    ViewBag.line = new { width = 0.0001 };
    return View();
}
public class LineChartData
{
    public DateTime xValue;
    public double yValue;
}

```

![Alt text](../images/step1.png)

Step 2: Now, add three striplines for three control limits: LCL, UCL, and CL.

CSHTML

```

@(Html.EJS().Chart("container").Title("Sales History of Product
X").Load("load").Loaded("loaded").PointRender("pointRender").Series(series
=>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
    XName("xValue").
    YName("yValue").
    Marker(mr => mr.Visible(true).Height(10).Width(10)).

```

```

        DataSource(ViewBag.dataSource).Add();
    }).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(mg =>
mg.Width(0)).MajorTickLines(mt => mt.Width(0)))
    .PrimaryYAxis(py => py.MajorGridLines(mg =>
mg.Width(0)).MajorTickLines(mt =>
mt.Width(0)).StripLines(ViewBag.yAxisStripLine))
    .Render()
)

```

STEP2.CS

```

public IActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(1975, 01, 01),
yValue = 36},
        new LineChartData { xValue = new DateTime(1980, 01, 01),
yValue = 32.5 },
        new LineChartData { xValue = new DateTime(1985, 01, 01),
yValue = 39 },
        new LineChartData { xValue = new DateTime(1990, 01, 01),
yValue = 34.4 },
        new LineChartData { xValue = new DateTime(1995, 01, 01),
yValue = 31.5 },
        new LineChartData { xValue = new DateTime(2000, 01, 01),
yValue = 40 },
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 60 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 66 },
    };
    ViewBag.dataSource = chartData;
    ViewBag.line = new { width = 0.0001 };
    List<ChartStripLine> yAxisStripLine = new
List<ChartStripLine>();
    ChartStripLine ystripline1 = new ChartStripLine();
    ystripline1.Start = "30";
    ystripline1.End = "40";
    ystripline1.Text = "";
    ystripline1.Color = "black";
    ystripline1.Visible = true;
    ChartStripLine ystripline2 = new ChartStripLine();
    ystripline2.Start = "20";
    ystripline2.End = "30";
    ystripline2.Text = "";
    ystripline2.Color = "black";
    ystripline2.Visible = true;
    ChartStripLine ystripline3 = new ChartStripLine();
    ystripline3.Start = "10";
    ystripline3.End = "20";
    ystripline3.Text = "";
    ystripline3.Color = "black";
    ystripline3.Visible = true;
}

```



```

        yAxisStripline.Add(ystripline1);
        yAxisStripline.Add(ystripline2);
        yAxisStripline.Add(ystripline3);
        ViewBag.yAxisStripLine = yAxisStripline;
        return View();
    }

    public class LineChartData
    {
        public DateTime xValue;
        public double yValue;
    }

```

Step 3: Then, add annotations to show the text for LCL, UCL, and CL in respective positions.

```
`html
```

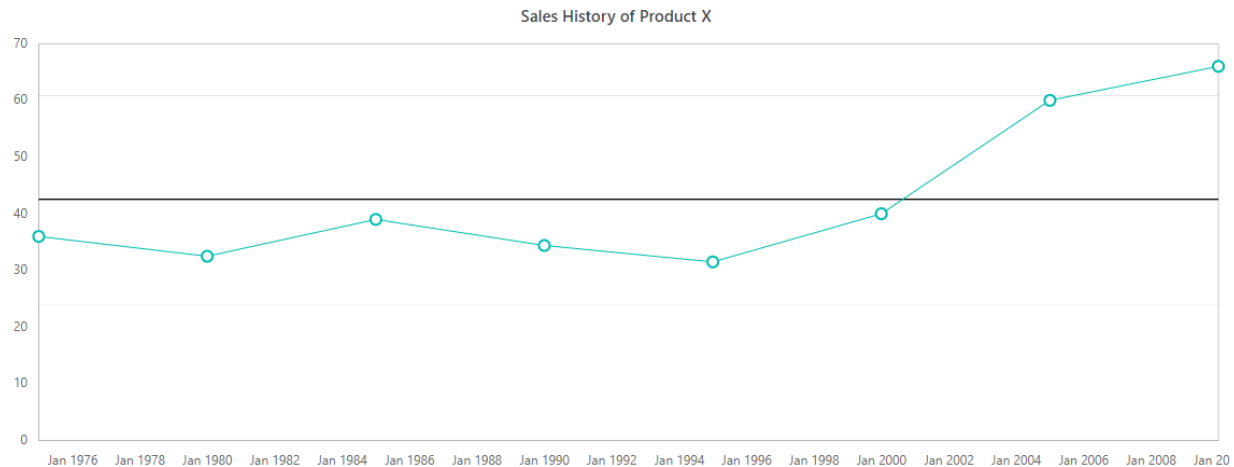
```

<ejs-chart id="container" title="Sales History of Product X">
  <e-chart-primaryxaxis valueType="DateTime" majorGridLines="ViewBag.line"
    majorTickLines="ViewBag.line"></e-chart-primaryxaxis>
  <e-chart-primaryyaxis striplines="ViewBag.yAxisStripLine" majorGridLines="ViewBag.line"
    majorTickLines="ViewBag.line"></e-chart-primaryyaxis>
  <e-series-collection>
    <e-series dataSource="ViewBag.dataSource" xName="xValue" yName="yValue"
      type="@Syncfusion.EJ2.Charts.ChartSeriesType.Line">
      <e-series-marker visible="true" height="10" width="10"></e-series-marker>
    </e-series>
  </e-series-collection>
  <e-chart-annotations>
    <e-chart-annotation Content="<div>LCL</div>" X="30" Y="2"
      CoordinateUnits='@Syncfusion.EJ2.Charts.Units.Point'></e-chart-annotation>
    <e-chart-annotation Content="<div>CL</div>" X="30" Y="2"
      CoordinateUnits='@Syncfusion.EJ2.Charts.Units.Point'></e-chart-annotation>
    <e-chart-annotation Content="<div>UCL</div>" X="30" Y="2"
      CoordinateUnits='@Syncfusion.EJ2.Charts.Units.Point'></e-chart-annotation>
    <e-chart-annotation Content="<div>Number of groups: xx</div><div>CL: xx</div><div>LCL:
      xx</div><div>UCL: xx</div><div>Standard Divation: xx</div>" X="400" Y="500"
      CoordinateUnits='@Syncfusion.EJ2.Charts.Units.Pixel'></e-chart-annotation>
  </e-chart-annotations>
</ejs-chart>
`

```

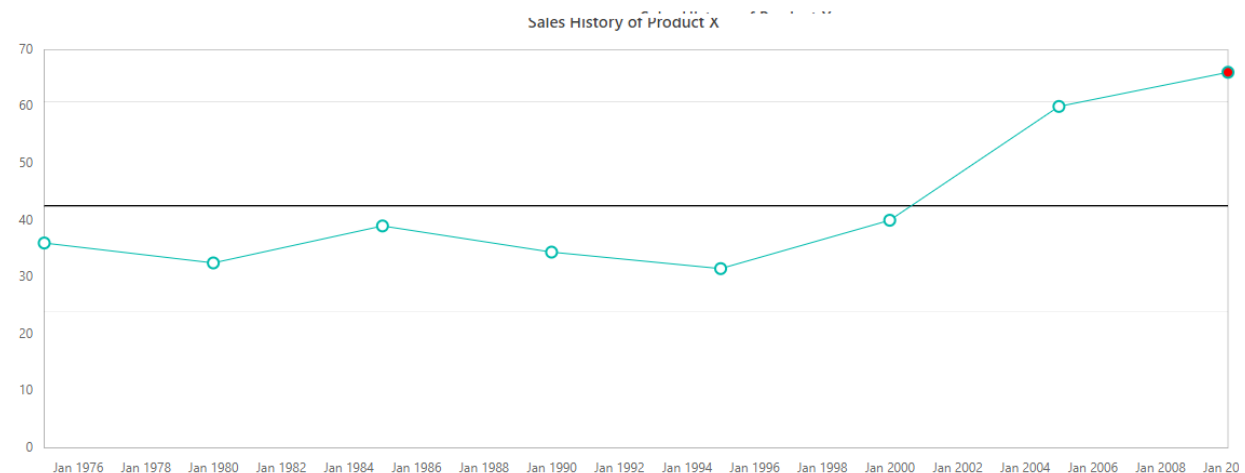
Step 4: Calculate the control limits based on the CL, LCL, and UCL. The start and end ranges have been set for strip lines in the load event.

```
`html
<script>
var mean = 0, LCL = 0, UCL = 0, s = 0;
function load(args) {
// calculate the mean value
for (var i = 0; i < args.chart.series[0].dataSource.length; i++) {
mean = (mean) + (+args.chart.series[0].dataSource[i].yValue);
}
mean = mean / args.chart.series[0].dataSource.length;
for (var i = 0; i < args.chart.series[0].dataSource.length; i++) {
s += ((+args.chart.series[0].dataSource[i].yValue) - mean) * ((+args.chart.series[0].dataSource[i].yValue) -
mean);
}
// calculate the standard divation here
s = s / args.chart.series[0].dataSource.length;
s = Math.sqrt(s);
// Calculate LCL and UCL
LCL = mean - (1.5 * s);
UCL = mean + (1.5 * s);
// set the strip line ranges based on the LCL and UCL
args.chart.primaryYAxis.stripLines[0].start = LCL;
args.chart.primaryYAxis.stripLines[0].end = LCL + 0.0125;
args.chart.primaryYAxis.stripLines[1].start = mean;
args.chart.primaryYAxis.stripLines[1].end = mean + 0.25;
args.chart.primaryYAxis.stripLines[2].start = UCL;
args.chart.primaryYAxis.stripLines[2].end = UCL + 0.0135;
}
</script>
`
```



Step 5: Highlight the points that are above the UCL using the `pointRender` event.

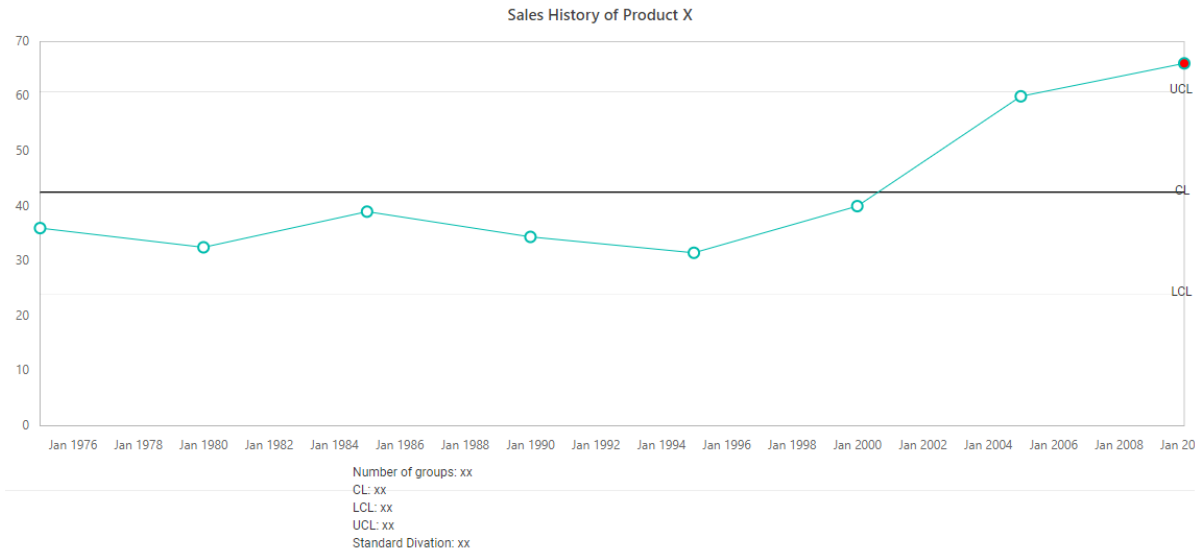
```
`html
<script>
function pointRender(args) {
  if (args.point.y > UCL) {
    args.fill = 'red';
  }
}
</script>
```



Step 6: Change the values of annotation text (UCL, LCL, mean, and CL).

```
`html
<script>
```

```
var mean = 0, LCL = 0, UCL = 0, s = 0;
function load(args) {
    // calculate the mean value
    for (var i = 0; i < args.chart.series[0].dataSource.length; i++) {
        mean = (mean) + (+args.chart.series[0].dataSource[i].yValue);
    }
    mean = mean / args.chart.series[0].dataSource.length;
    for (var i = 0; i < args.chart.series[0].dataSource.length; i++) {
        s += ((+args.chart.series[0].dataSource[i].yValue) - mean) * ((+args.chart.series[0].dataSource[i].yValue) - mean);
    }
    // calculate the standard divation here
    s = s / args.chart.series[0].dataSource.length;
    s = Math.sqrt(s);
    // Calculate LCL and UCL
    LCL = mean - (1.5 * s);
    UCL = mean + (1.5 * s);
    // set the annotations x and y positions based on the LCL and UCL
    args.chart.annotations[0].x = args.chart.series[0].dataSource[args.chart.series[0].dataSource.length - 1].xValue;
    args.chart.annotations[0].y = LCL;
    args.chart.annotations[1].x = args.chart.series[0].dataSource[args.chart.series[0].dataSource.length - 1].xValue;
    args.chart.annotations[1].y = mean;
    args.chart.annotations[2].x = args.chart.series[0].dataSource[args.chart.series[0].dataSource.length - 1].xValue;
    args.chart.annotations[2].y = UCL;
}
</script>
```



Step 7: Add the dasharray for stripline elements and add some pixel for the annotation elements to move from its default position and then add the text for the last annotation to show the mean, UCL, CL, LCL and standard deviation in the loaded event.

```
`html
<script>
function loaded(args) {
// Add dasharray for strip line elements
var strip1 = document.getElementById('containerstriplineBehindrect0');
strip1.setAttribute('stroke', 'black');
strip1.setAttribute('stroke-dasharray', '5');
var strip2 = document.getElementById('containerstriplineBehindrect2');
strip2.setAttribute('stroke', 'black');
strip2.setAttribute('stroke-dasharray', '5');
var annotation = document.getElementById('containerAnnotationCollections');
var child, left;
// move the annotation elements to nearest chart area by adding some pixel to annotation elements
for (var i = 0; i < annotation.children.length - 1; i++) {
child = annotation.children[i];
left = child.style.left;
left = +(left.replace('px', '')) + 20;
child.style.left = left + 'px';
}
}
```

```
// Add the annotation text to show the mean, UCL, CL, LCL and standard divation
annotation.children[3].children[0].innerText = annotation.children[3].children[0].innerText.replace('xx',
args.chart.series[0].dataSource.length);

annotation.children[3].children[1].innerText = annotation.children[3].children[1].innerText.replace('xx',
mean);

annotation.children[3].children[2].innerText = annotation.children[3].children[2].innerText.replace('xx',
LCL);

annotation.children[3].children[3].innerText = annotation.children[3].children[3].innerText.replace('xx',
UCL);

annotation.children[3].children[4].innerText = annotation.children[3].children[4].innerText.replace('xx',
s);
}
</script>
`
```

CSHTML

```
@(Html.EJS().Chart("container").Title("Sales History of Product
X").Load("load").Loaded("loaded").PointRender("pointRender").Series(series
=>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
        XName("xValue").
        YName("yValue").
        Marker(mr => mr.Visible(true).Height(10).Width(10)).
        DataSource(ViewBag.dataSource).Add();
}).Annotations(an =>
{
    an.X("30").Y("2").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Content
        ("<div>LCL</div>").Add();

    an.X("30").Y("2").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Content
        ("<div>CL</div>").Add();

    an.X("30").Y("2").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Content
        ("<div>UCL</div>").Add();

    an.X("400").Y("500").CoordinateUnits(Syncfusion.EJ2.Charts.Units.Point).Cont
ent("<div>Number of groups: xx</div><div>CL: xx</div><div>LCL:
xx</div><div>UCL: xx</div><div>Standard Divation: xx</div>").Add();
}).
    PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).MajorGridLines(mg =>
mg.Width(0)).MajorTickLines(mt => mt.Width(0))
    .PrimaryYAxis(py => py.MajorGridLines(mg =>
mg.Width(0)).MajorTickLines(mt =>
mt.Width(0)).StripLines(ViewBag.yAxisStripLine))
    .Render()
)
<script>
```

```

var mean = 0, LCL = 0, UCL = 0, s = 0;
function load(args) {
    // calculate the mean value
    for (var i = 0; i < args.chart.series[0].dataSource.length; i++) {
        mean = (mean) + (+args.chart.series[0].dataSource[i].yValue);
    }
    mean = mean / args.chart.series[0].dataSource.length;
    for (var i = 0; i < args.chart.series[0].dataSource.length; i++) {
        s += ((+args.chart.series[0].dataSource[i].yValue) - mean) *
        ((+args.chart.series[0].dataSource[i].yValue) - mean);
    }
    // calculate the standard divation here
    s = s / args.chart.series[0].dataSource.length;
    s = Math.sqrt(s);
    // Calculate LCL and UCL
    LCL = mean - (1.5 * s);
    UCL = mean + (1.5 * s);
    // set the strip line ranges based on the LCL and UCL
    args.chart.primaryYAxis.stripLines[0].start = LCL;
    args.chart.primaryYAxis.stripLines[0].end = LCL + 0.0125;
    args.chart.primaryYAxis.stripLines[1].start = mean;
    args.chart.primaryYAxis.stripLines[1].end = mean + 0.25;
    args.chart.primaryYAxis.stripLines[2].start = UCL;
    args.chart.primaryYAxis.stripLines[2].end = UCL + 0.0135;
    args.chart.annotations[0].x =
args.chart.series[0].dataSource[args.chart.series[0].dataSource.length -
1].xValue;
    args.chart.annotations[0].y = LCL;
    args.chart.annotations[1].x =
args.chart.series[0].dataSource[args.chart.series[0].dataSource.length -
1].xValue;
    args.chart.annotations[1].y = mean;
    args.chart.annotations[2].x =
args.chart.series[0].dataSource[args.chart.series[0].dataSource.length -
1].xValue;
    args.chart.annotations[2].y = UCL;
}
function loaded(args) {
    var stripl =
document.getElementById('container_stripline_Behind_rect_primaryYAxis_0');
    stripl.setAttribute('stroke', 'black');
    stripl.setAttribute('stroke-dasharray', '5');
    var stripl2 =
document.getElementById('container_stripline_Behind_rect_primaryYAxis_0');
    stripl2.setAttribute('stroke', 'black');
    stripl2.setAttribute('stroke-dasharray', '5');
    var annotation =
document.getElementById('container_Annotation_Collections');
    var child, left;
    for (var i = 0; i < annotation.children.length - 1; i++) {
        child = annotation.children[i];
        left = child.style.left;
        left = +(left.replace('px', '')) + 20;
        child.style.left = left + 'px';
    }
}

```

```

        annotation.children[3].children[0].innerText =
annotation.children[3].children[0].innerText.replace('xx',
args.chart.series[0].dataSource.length);
        annotation.children[3].children[1].innerText =
annotation.children[3].children[1].innerText.replace('xx', mean);
        annotation.children[3].children[2].innerText =
annotation.children[3].children[2].innerText.replace('xx', LCL);
        annotation.children[3].children[3].innerText =
annotation.children[3].children[3].innerText.replace('xx', UCL);
        annotation.children[3].children[4].innerText =
annotation.children[3].children[4].innerText.replace('xx', s);
    }
    function pointRender(args) {
        if (args.point.y > UCL) {
            args.fill = 'red';
        }
    }
}
</script>

```

CONTROL.CS

```

public IActionResult Index()
{
    List<LineChartData> chartData = new List<LineChartData>
    {
        new LineChartData { xValue = new DateTime(1975, 01, 01),
yValue = 36},
        new LineChartData { xValue = new DateTime(1980, 01, 01),
yValue = 32.5 },
        new LineChartData { xValue = new DateTime(1985, 01, 01),
yValue = 39 },
        new LineChartData { xValue = new DateTime(1990, 01, 01),
yValue = 34.4 },
        new LineChartData { xValue = new DateTime(1995, 01, 01),
yValue = 31.5 },
        new LineChartData { xValue = new DateTime(2000, 01, 01),
yValue = 40 },
        new LineChartData { xValue = new DateTime(2005, 01, 01),
yValue = 60 },
        new LineChartData { xValue = new DateTime(2010, 01, 01),
yValue = 66 },
    };
    ViewBag.dataSource = chartData;

    List<ChartStripLine> yAxisStripline = new
List<ChartStripLine>();
    ChartStripLine ystripline1 = new ChartStripLine();
    ystripline1.Start = "30";
    ystripline1.End = "40";
    ystripline1.Text = "";
    ystripline1.Color = "black";
    ystripline1.Visible = true;
    ChartStripLine ystripline2 = new ChartStripLine();
    ystripline2.Start = "20";
    ystripline2.End = "30";
    ystripline2.Text = "";
}

```

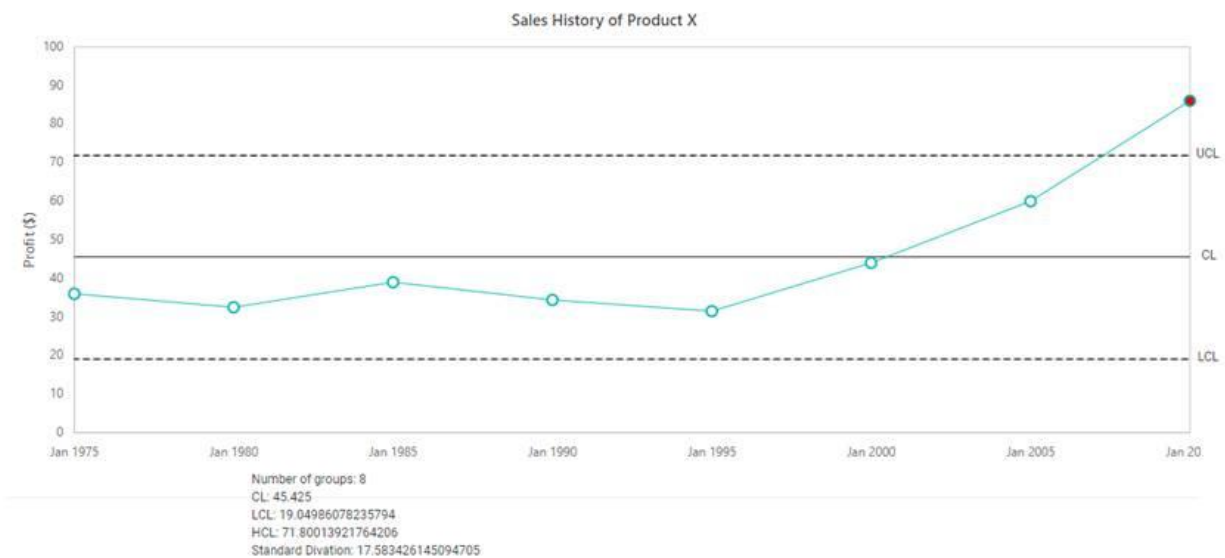


```

ystripline2.Color = "black";
ystripline2.Visible = true;
ChartStripLine ystripline3 = new ChartStripLine();
ystripline3.Start = "10";
ystripline3.End = "20";
ystripline3.Text = "";
ystripline3.Color = "black";
ystripline3.Visible = true;
yAxisStipline.Add(ystripline1);
yAxisStipline.Add(ystripline2);
yAxisStipline.Add(ystripline3);
ViewBag.yAxisStripLine = yAxisStipline;
return View();
}
public class LineChartData
{
    public DateTime xValue;
    public double yValue;
}

```

Screenshots:



Sample link: A sample for creating control chart is available in the following link, [chartSample](#).

<!-- markdownlint-disable MD036 -->

Add or remove a series from the chart dynamically

You can add or remove the chart series dynamically by using the `addSeries` or `removeSeries` method.

To add or remove the series dynamically, follow the given steps:

Step 1:

To add a new series to chart dynamically, pass the series value to the `addSeries` method.

To remove the new series from chart dynamically, pass the series index to the `removeSeries` method.

CSHTML

```

<button id="add">Add</button>
<button id="remove">Remove</button>
@Html.EJS().Chart("container").PrimaryXAxis(px => px.Title("Years").
ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Series(series =>
{
    series.Type(
        Syncfusion.EJ2.Charts.ChartSeriesType.Column
    ).DataSource(ViewBag.dataSource).Name("Product
X").XName("x").YName("yValue").Add();
}
).PrimaryYAxis(py => py.Title("Profit
($)")).Title("Sales History of Product X").Render()
<script>
    document.getElementById("add").onclick = function () {
        var chart = document.getElementById("container").ej2_instances[0];
        chart.addSeries([
            {
                type: 'Column',
                dataSource: [
                    { x: 'John', y: 11000 }, { x: 'Jake', y: 16000 }, {
x: 'Peter', y: 19000 },
                    { x: 'James', y: 12000 }, { x: 'Mary', y: 10700 }],
                xName: 'x', width: 2,
                yName: 'y'
            }
        ]);
    };
    document.getElementById("remove").onclick = function () {
        var chart = document.getElementById("container").ej2_instances[0];
        chart.removeSeries(1);
    }
</script>

```

ADD-REMOVE.CS

```

public ActionResult Index()
{
    List<ColumnChartData> chartData = new List<ColumnChartData>
    {
        new ColumnChartData { x= "John", yValue= 10000, yValue1=37,
yValue2=38 },
        new ColumnChartData { x= "Jake", yValue= 12000, yValue1=23,
yValue2=17 },
        new ColumnChartData { x= "Peter", yValue= 18000, yValue1=18,
yValue2=26 },
        new ColumnChartData { x= "James", yValue= 11000, yValue1=18,
yValue2=26 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnChartData
{
    public string x;
    public double yValue;
    public double yValue1;
    public double yValue2;
}

```

```
<!-- markdownlint-disable MD036 -->
```

Render chart from code behind and update the chart using partial view

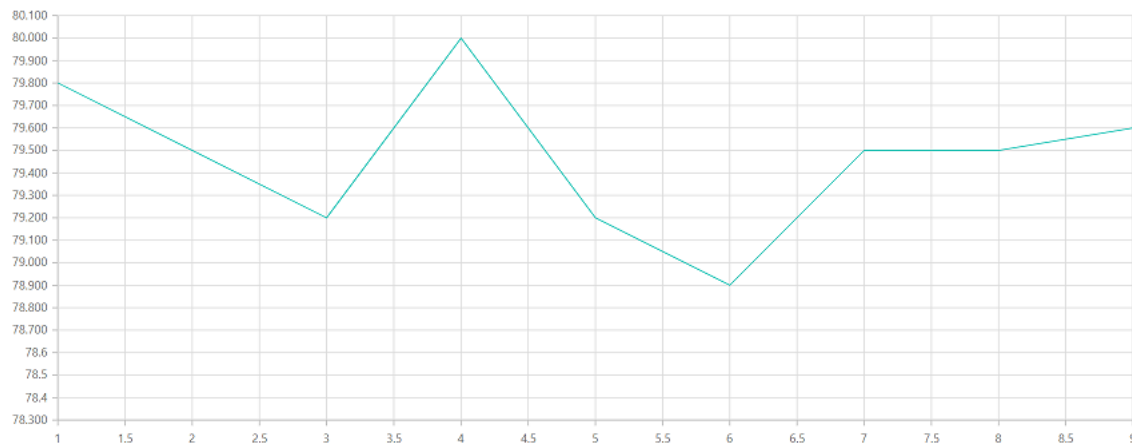
You can render the charts from code behind and update the charts in partial view by calling the partial view action method in ajax.

Follow the given steps to render and update the charts from code behind.

Step 1: Initially, render the home view page, and then call the partial view action method via ajax call in the home view page. In that chart is rendered using client side content.

```
`html
<div class="row" style="margin-top:10px">
<button onclick="refreshChart()">Update Chart</button>
</div>
<div class="row" id="container">
@Html.Partial("_ViewChart");
</div>
<script>
function refreshChart() {
var ajax = new ej.base.Ajax();
ajax.url = "@Url.Action("Create")";
ajax.type = "GET";
ajax.successHandler = function (args) {
$('#container').html(args);
}
ajax.send();
}
</script>
`
```

Initial Chart



Step 2: Then create chart model properties in controller page. After initial rendering then update the created chart properties in the home view page through partial view action method in ajax call.

```
`cs
// partial view action method
[HttpGet]
public ActionResult Create()
{
    Chart chartModel = new Chart();
    InitializeChart(chartModel);
    ViewData["ChartModel"] = chartModel;
    return PartialView("_ViewChart");
}

private void InitializeChart(Chart chartModel)
{
    List<LineChartData> data = new List<LineChartData>();
    data.Add(new LineChartData(10, 12, 26));
    data.Add(new LineChartData(20, 14, 22));
    data.Add(new LineChartData(30, 16, 18));
    data.Add(new LineChartData(40, 13, 35));
    data.Add(new LineChartData(50, 11, 24));
    ChartSeries series1 = new ChartSeries();
    series1.XName = "Xvalue";
    series1.YName = "YValue1";
    series1.DataSource = data;
    series1.Fill = "red";
}
```

```
series1.Name = "Series1";
series1.Type = Syncfusion.EJ2.Charts.ChartSeriesType.Column;
series1.Trendlines = new List<ChartTrendline>();
series1.Segments = new List<ChartSegment>();
series1.Visible = true;
ChartSeries series2 = new ChartSeries();
series2.XName = "Xvalue";
series2.YName = "yValue2";
series2.DataSource = data;
series2.Fill = "blue";
series2.Name = "Series2";
series2.Type = Syncfusion.EJ2.Charts.ChartSeriesType.Column;
series2.Trendlines = new List<ChartTrendline>();
series2.Segments = new List<ChartSegment>();
series2.Visible = true;
chartModel.Series = new List<ChartSeries>();
chartModel.PrimaryXAxis = new ChartAxis();
chartModel.PrimaryXAxis.Minimum = 5;
chartModel.PrimaryXAxis.Maximum = 50;
chartModel.PrimaryXAxis.Interval = 2;
chartModel.Series.Add(series1);
chartModel.Series.Add(series2);
}
}
[Serializable]
public class LineChartData
{
    public LineChartData(double xval, double yvalue1, double yvalue2)
    {
        this.Xvalue = xval;
        this.YValue1 = yvalue1;
        this.yValue2 = yvalue2;
    }
}
```

```
public double Xvalue { get; set; }
public double YValue1 { get; set; }
public double yValue2 { get; set; }
}
`
```

Step 3: Return the partial view page when calling partial view action method. In that partial view, refer the `scriptManager` to render the charts. Using `viewData` you can get chart object in view page from code behind and append it to chart container.

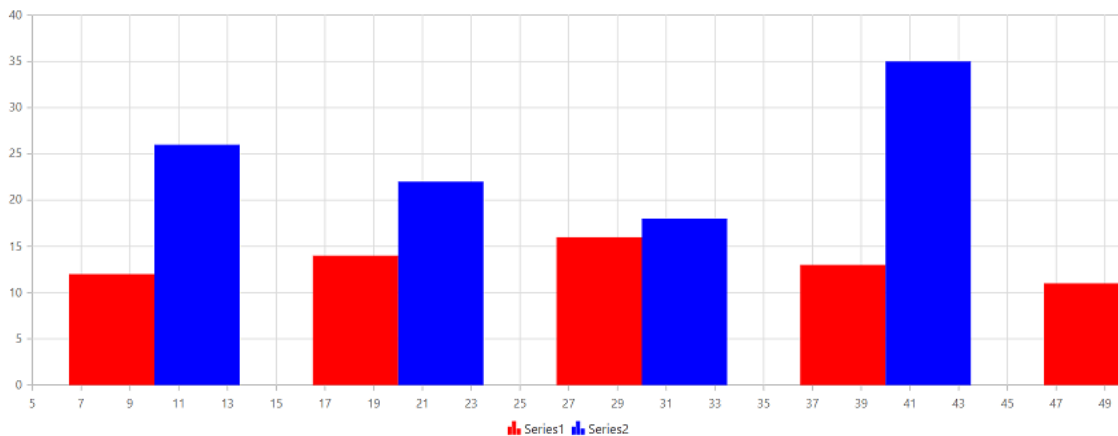
```
`html
<div>
@if ((Syncfusion.EJ2.Charts.Chart)ViewData["ChartModel"] == null)
{
    @Html.EJS().Chart("container").Load("firstChartLoad").Render();
}
else
{
    @Html.EJS().Chart("container", (Syncfusion.EJ2.Charts.Chart)ViewData["ChartModel"]).Render();
}
</div>
<script>
var firstChartLoad = function (args) {
    args.chart.series[0].dataSource = getData();
    args.chart.series[0].xName = "x";
    args.chart.series[0].yName = "y";
}
function getData() {
    let data = [];
    let point;
    let value = 80;
    for (let i = 1; i < 10; i++) {
        if (Math.random() > .5) {
            value += Math.random();
        } else {
            value -= Math.random();
        }
    }
}
```

```

}
point = { x: i, y: value.toFixed(1) };
data.push(point);
}
return data;
}
</script>
@Html.EJS().ScriptManager()
`

```

Updated Chart



Sample reference

Sample for how to render the charts from code behind and update using partial view [chart sample](#)

To add chart dynamically

By using html button, you can add the chart dynamically by clicking the button.

To add the chart dynamically through button click, follow the given steps:

Step 1: Initially create the html button. Then create chart inside of button `onClick` function. Now, clicking the button charts will render based on click count.

CSHTML

```

<button id="add" onclick="addChart()">Add Chart</button>
<script>
    window.count = 0;
    function addChart() {
        //Create div element dynamically and append to DOM
        var chartEle = document.createElement('div');
        chartEle.id = 'chartContainer' + window.count;
        document.getElementsByTagName('body')[0].appendChild(chartEle);
        //Created chart here
        var chart = new ej.charts.Chart({
            series: [{

```

```

        type: 'Line', xName: 'x', width: 2, marker: { visible: true
    },
        yName: 'y', name: 'Germany',
        dataSource: [{ x: 1, y: 21 }, { x: 2, y: 24 }, { x: 3, y: 36
    },
        { x: 4, y: 38 }, { x: 5, y: 54 }, { x: 6, y: 57 }, { x:
    7, y: 70 }],
        title: 'Inflation - Consumer Price', tooltip: { enable: true },
        height: '400', width: '800'
    });
    chart.appendTo('#' + chartEle.id);
    window.count++;
}
</script>

```

DYNAMIC-CHART.CS

```

public IActionResult Index()
{
    return View();
}

```

3D Chart

Getting started with ASP.NET MVC 3D Chart control

This section briefly explains about how to include [ASP.NET MVC 3D Chart](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Views/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{ site.ej2version
}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Views/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC 3D Chart control

Now, add the Syncfusion ASP.NET MVC 3D Chart control in **~/Home/Index.cshtml** page.

CSHTML

```
@model List<ThreeDimensionalChartSample.Controllers.Data>

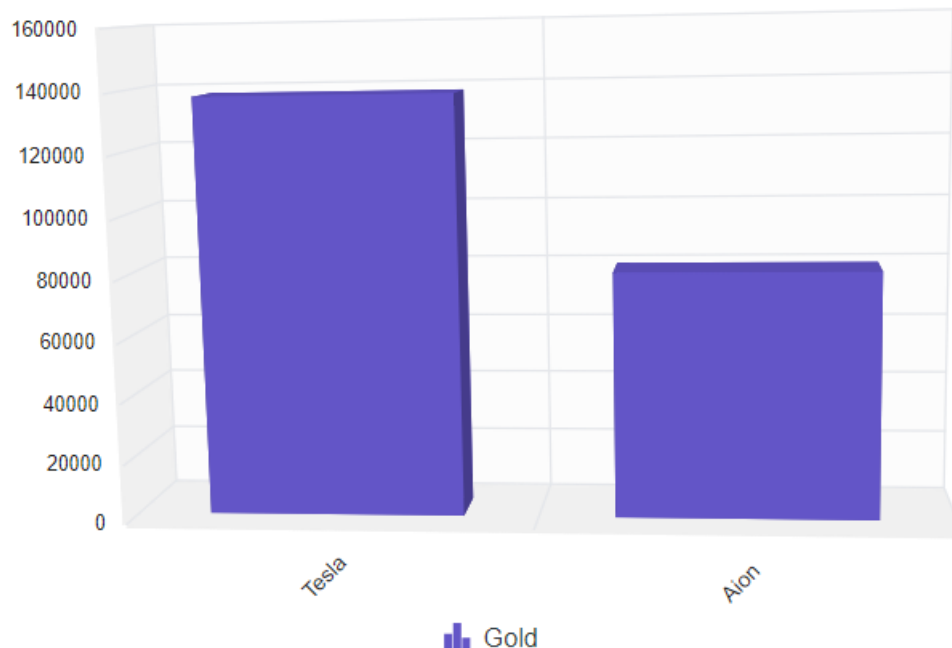
@Html.EJS().Chart3D("container").EnableRotation(true).Rotation(7).Tilt(10).Depth(100).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
).Series(series =>
```

```
{  
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).DataSource(Model  
    ).XName("X").YName("Y").Name("Gold").Add();  
}).Render();
```

GETTING-STARTED.CS

```
public IActionResult Index()  
{  
    List<Data> data = new List<Data>  
    {  
        new Data { X= "Tesla", Y= 137429 },  
        new Data { X= "Aion", Y= 80308 }  
    };  
    ViewBag.dataSource = data;  
    return View();  
}  
public class Data  
{  
    public string X;  
    public double Y;  
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC 3D Chart control will be rendered in the default web browser.



Note: View Sample in GitHub.

Working with data in ASP.NET MVC 3D Chart Component

Local data

A simple JSON data can be bound to the 3D chart using [DataSource](#) property in series. Now map the fields in JSON to [XName](#) and [YName](#) properties.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    ).Render())
```

LOCAL-DATA.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}
```

Remote data

The remote data can be bound to the 3D chart using the **DataManager**. The **DataManager** requires minimal information like web service URL, adaptor and cross domain to interact with service endpoint properly. Assign the instance of the **DataManager** to the **DataSource** property in series and map the fields of data to **XName** and **YName** properties. You can also use the **Query** property of the series to filter the data.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("CustomerID").
        YName("Freight").
        DataSource("dataManager").
        Query("query").
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    ).Render())
<script>
    var dataManager = new ej.data.DataManager({
        url: 'https://services.syncfusion.com/aspnet/production/api/orders'
    });
    var query = new ej.data.Query().take(5).where('Estimate', 'lessThan', 3,
false);
</script>
```

REMOTE-DATA.CS

```
public ActionResult Index()
{
    return View();
}
```

Binding data using ODataAdaptor

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the **DataManager**. Refer to the following code example for remote Data binding using OData service.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("CustomerID").
```

```

        YName("Freight").
        DataSource("dataManager").
        Query("query").
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    ).Render()
<script>
    var dataManager = new ej.data.DataManager({
        url: 'https://services.syncfusion.com/aspnet/production/api/orders',
        adaptor: new ej.data.ODataAdaptor()
    });
    var query = new ej.data.Query();
</script>

```

ADAPTOR.CS

```

public ActionResult Index()
{
    return View();
}

```

Empty points

The data points that uses the **null** or **undefined** as value are considered as empty points. The empty data points are ignored and is not plotted in the chart. When the data is provided by using the points property, by using [EmptyPointSettings](#) property in series, the empty can be customized. The default [Mode](#) of the empty point is **Gap**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        EmptyPointSettings(em =>
        em.Mode(Syncfusion.EJ2.Charts.EmptyPointMode.Gap)).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    ).Render()

```

EMPTY-POINTS.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= double.NaN },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= double.NaN },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}

```

Customizing empty point

The specific color for empty point can be set by the [Fill](#) property in [EmptyPointSettings](#).

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("month").
YName("sales").
EmptyPointSettings(em =>
em.Mode(Syncfusion.EJ2.Charts.EmptyPointMode.Average).Fill("green")).
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
).Render())

```

CUSTOM-EMPTYPOINT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>

```

```

{
    new ChartData { month= "Jan", sales= 35 },
    new ChartData { month= "Feb", sales= 28 },
    new ChartData { month= "Mar", sales= double.NaN },
    new ChartData { month= "Apr", sales= 32 },
    new ChartData { month= "May", sales= 40 },
    new ChartData { month= "Jun", sales= 32 },
    new ChartData { month= "Jul", sales= 35 },
    new ChartData { month= "Aug", sales= double.NaN },
    new ChartData { month= "Sep", sales= 38 },
    new ChartData { month= "Oct", sales= 30 },
    new ChartData { month= "Nov", sales= 25 },
    new ChartData { month= "Dec", sales= 32 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ChartData
{
    public string month;
    public double sales;
}

```

Dimensions in ASP.NET MVC 3D Chart Component

Size for container

The 3D chart can be rendered to its container size and it can be set via inline or CSS as demonstrated below.

CSHTML

```

@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100).Width("650").Height("350")
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render())

```

SIZE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
    }
}

```

```

        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string month;
    public double sales;
}

```

Size for chart

<!-- markdownlint-disable MD036 -->

The size of the 3D chart can be set directly through [Width](#) and [Height](#) properties.

In pixel

The size of the 3D chart can be set in pixel as demonstrated below.

CSHTML

```

@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100).Width("650px").Height("350px")
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render())

```

PIXEL.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
    }
}

```



```

        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string month;
    public double sales;
}

```

In percentage

By setting the value in percentage, 3D chart gets its dimension with respect to its container. For example, when the height is **50%**, chart renders to half of the container height.

CSHTML

```

@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100).Width("80%").Height("90%")
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render())

```

PERCENTAGE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    }
}

```

```

    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string month;
    public double sales;
}

```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Category axis in ASP.NET MVC 3D Chart Component

The category axis is the horizontal axis of a 3D chart that shows text values rather than numerical values. Compared to the vertical axis, this axis has fewer labels. The following sample shows to render the 3D chart using category axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render())

```

CATEGORY.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
}

```

```
public double gold;
}
```

Labels placement

By default, category axis labels are placed between ticks in an axis. It can also be placed on ticks using the [LabelPlacement](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.OnTicks)
    ).Title("Olympic Medals")
    .Render())
```

LABELPLACEMENT.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA", gold= 50 },
        new ChartData { country= "China", gold= 40 },
        new ChartData { country= "Japan", gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France", gold= 50 },
        new ChartData { country= "Germany", gold= 40 },
        new ChartData { country= "Italy", gold= 40 },
        new ChartData { country= "Sweden", gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}
```

Range

The range of the category axis can be customized using [Minimum](#), [Maximum](#) and [Interval](#) properties of the axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Minimum(1).Maximum(5)
    .Interval(2)
    ).Title("Olympic Medals")
    .Render())
```

RANGE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Indexed category axis

The category axis can also be rendered based on the index values of the data source. This can be achieved by defining the [IsIndexed](#) property to **true** in the axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
```

```

        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).IsIndexed(true)
    ).Title("Olympic Medals")
    .Render()

```

INDEX.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70 },
        new ChartData { country= "China",    gold= 40, silver= 60 },
        new ChartData { country= "Japan",    gold= 70, silver= 60 },
        new ChartData { country= "Australia", gold= 60, silver= 56 },
        new ChartData { country= "France",   gold= 50, silver= 45 },
        new ChartData { country= "Germany",  gold= 40, silver= 30 },
        new ChartData { country= "Italy",    gold= 40, silver= 35 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
}

```

Numeric axis in ASP.NET MVC 3D Chart Component

The **Numeric Axis** can be used to represent the numeric values of data in 3D chart. By default, the [ValueType](#) of an axis is **Double**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })

```

```
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
)
.Render()
```

DOUBLE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
}
```

Range

The range of an axis will be calculated automatically based on the provided data, and it can also be customized by using the [Minimum](#), [Maximum](#) and [Interval](#) properties of the axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("x").
    YName("y").
    ColumnSpacing(0.1).
})
```

```

        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double).Minimum(1).Maximum(20).
Interval(5)
)
.Render())

```

RANGE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
}

```

Range padding

Padding can be applied to the minimum and maximum extremes of an axis range by using the [RangePadding](#) property. Numeric axis supports the following types of padding.

- None
- Round
- Additional
- Normal

- Auto

Numeric - None

When the [RangePadding](#) is set to **None**, minimum and maximum of the axis is based on the data.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
    )
    .PrimaryYAxis(py =>
        py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    )
    .Render())
```

NONE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
```



```
public class ChartData
{
    public double x;
    public double y;
}
```

Numeric - Round

When the [RangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value, which is divisible by interval. For example, when the [Minimum](#) is 3.5 and the [Interval](#) is 1, then the minimum will be rounded to 3.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
    )
    .PrimaryYAxis(py =>
        py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Round)
    )
    .Render())
```

ROUND.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
    }
```

```

        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public double x;
    public double y;
}

```

Numeric - Additional

When the [RangePadding](#) is set to **Additional**, interval of an axis will be added to the minimum and maximum of the axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
    )
    .PrimaryYAxis(px =>
        px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional)
    )
    .Render())

```

ADDITIONAL.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
    }
}

```

```

        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public double x;
    public double y;
}

```

Numeric - Normal

When the [RangePadding](#) is set to **Normal**, padding is applied to the axis based on default range calculation.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
    )
    .PrimaryYAxis(py =>
        py.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Normal)
    )
    .Render())

```

NORMAL.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
    }
}

```

```

        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16},
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public double x;
    public double y;
}

```

Numeric - Auto

When the [RangePadding](#) is set to **Auto**, horizontal numeric axis takes **None** as padding calculation, while the vertical numeric axis takes **Normal** as padding calculation.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
        .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Auto)
    )
    .PrimaryYAxis(px =>
        px.RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Auto)
    )
    .Render())

```

AUTO.CS

```

public ActionResult Index()
{

```

```

List<ChartData> chartData = new List<ChartData>
{
    new ChartData { x= 1, y= 7 },
    new ChartData { x= 2, y= 1 },
    new ChartData { x= 3, y= 1 },
    new ChartData { x= 4, y= 14 },
    new ChartData { x= 5, y= 1 },
    new ChartData { x= 6, y= 10 },
    new ChartData { x= 7, y= 8 },
    new ChartData { x= 8, y= 6 },
    new ChartData { x= 9, y= 10 },
    new ChartData { x= 10, y= 10 },
    new ChartData { x= 11, y= 16 },
    new ChartData { x= 12, y= 6 },
    new ChartData { x= 13, y= 14 },
    new ChartData { x= 14, y= 7 },
    new ChartData { x= 15, y= 5 },
    new ChartData { x= 16, y= 2 },
    new ChartData { x= 17, y= 14 },
    new ChartData { x= 18, y= 7 },
    new ChartData { x= 19, y= 7 },
    new ChartData { x= 20, y= 10 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ChartData
{
    public double x;
    public double y;
}

```

Label format

Numeric label format

Numeric labels can be formatted by using the [LabelFormat](#) property. Also, it supports all globalize format.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Double)
    )
    .PrimaryYAxis(py =>
        py.LabelFormat("c")
    )

```

```
)
.Render()
```

LABEL-FORMAT.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
}
```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place.
1000	n2	1000.00	The Number is rounded to 2 decimal place.
1000	n3	1000.000	The Number is rounded to 3 decimal place.
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place.

0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place.

Grouping separator

To separate the y-axis labels to groups of thousands, set the [UseGroupingSeparator](#) property to **true** in the 3D chart.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .UseGroupingSeparator(true)
    .Render())
```

GROUP-SEPARATOR.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 10, y= 7000 },
        new ChartData { x= 20, y= 1000 },
        new ChartData { x= 30, y= 12000 },
        new ChartData { x= 40, y= 14000 },
        new ChartData { x= 50, y= 11000 },
        new ChartData { x= 60, y= 5000 },
        new ChartData { x= 70, y= 7300 },
        new ChartData { x= 80, y= 9000 },
        new ChartData { x= 90, y= 12000 },
        new ChartData { x= 100, y= 14000 },
        new ChartData { x= 110, y= 11000 },
        new ChartData { x= 120, y= 5000 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
```

```
{
    public double x;
    public double y;
}
```

Custom label format

The axis supports custom label format using placeholder like **{value}°C**, in which the value represent the axis label e.g 20°C.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryYAxis(py =>
        py.LabelFormat("${value}K")
    )
    .Render())
```

CUSTOM-FORMAT.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 1, y= 7 },
        new ChartData { x= 2, y= 1 },
        new ChartData { x= 3, y= 1 },
        new ChartData { x= 4, y= 14 },
        new ChartData { x= 5, y= 1 },
        new ChartData { x= 6, y= 10 },
        new ChartData { x= 7, y= 8 },
        new ChartData { x= 8, y= 6 },
        new ChartData { x= 9, y= 10 },
        new ChartData { x= 10, y= 10 },
        new ChartData { x= 11, y= 16 },
        new ChartData { x= 12, y= 6 },
        new ChartData { x= 13, y= 14 },
        new ChartData { x= 14, y= 7 },
        new ChartData { x= 15, y= 5 },
        new ChartData { x= 16, y= 2 },
        new ChartData { x= 17, y= 14 },
        new ChartData { x= 18, y= 7 },
        new ChartData { x= 19, y= 7 },
        new ChartData { x= 20, y= 10 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
```



```

}
public class ChartData
{
    public double x;
    public double y;
}

```

DateTime axis in ASP.NET MVC 3D Chart Component

DateTime axis

DateTime axis uses date time scale and displays the date time values as axis labels in the specified format.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
)
.Render())

```

DATETIME.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2000, 07, 11), y= 10 },
        new ChartData { x= new DateTime(2002, 04, 07), y= 30 },
        new ChartData { x= new DateTime(2004, 04, 06), y= 15 },
        new ChartData { x= new DateTime(2006, 04, 30), y= 65 },
        new ChartData { x= new DateTime(2008, 04, 08), y= 90 },
        new ChartData { x= new DateTime(2010, 04, 08), y= 85 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public DateTime x;
    public double y;
}

```

DateTime category axis

DateTime category axis is used to display the date time values with non-linear intervals. For example, the business days alone have been depicted in a week here.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTimeCategory).Skeleton("Ed"
)
)
.Render())
```

DATETIME-CATEGORY.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2017, 12, 20), y= 21 },
        new ChartData { x= new DateTime(2017, 12, 21), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 22), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 26), y= 70 },
        new ChartData { x= new DateTime(2017, 12, 27), y= 75 },
        new ChartData { x= new DateTime(2018, 01, 02), y= 82 },
        new ChartData { x= new DateTime(2018, 01, 03), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 04), y= 54 },
        new ChartData { x= new DateTime(2018, 01, 05), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 08), y= 45 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}
```

Range

Range of an axis will be calculated automatically based on the provided data. You can also customize the range of an axis using [Minimum](#), [Maximum](#) and [Interval](#) properties.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .Minimum(new DateTime(2000, 07, 01))
    .Maximum(new DateTime(2010, 02, 01))
    .Interval(1)
    )
    .Render())
```

RANGE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2000, 07, 11), y= 10 },
        new ChartData { x= new DateTime(2002, 04, 07), y= 30 },
        new ChartData { x= new DateTime(2004, 04, 06), y= 15 },
        new ChartData { x= new DateTime(2006, 04, 30), y= 65 },
        new ChartData { x= new DateTime(2008, 04, 08), y= 90 },
        new ChartData { x= new DateTime(2010, 04, 08), y= 85 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public DateTime x;
    public double y;
}
```

Interval customization

Date time intervals can be customized by using the [Interval](#) and [IntervalType](#) properties of the axis. For example, when you set **Interval** as **2** and **IntervalType** as **Years**, it considers 2 years as interval. DateTime axis supports following interval types,

- Auto
- Years
- Months
- Days
- Hours
- Minutes
- Seconds

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .IntervalType(Syncfusion.EJ2.Charts.IntervalType.Years)
    )
    .Render())
```

INTERVAL.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2000, 07, 11), y= 10 },
        new ChartData { x= new DateTime(2002, 04, 07), y= 30 },
        new ChartData { x= new DateTime(2004, 04, 06), y= 15 },
        new ChartData { x= new DateTime(2006, 04, 30), y= 65 },
        new ChartData { x= new DateTime(2008, 04, 08), y= 90 },
        new ChartData { x= new DateTime(2010, 04, 08), y= 85 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}
```

Applying padding to the range

Padding can be applied to the minimum and maximum extremes of the range by using the [RangePadding](#) property. DateTime axis supports the following types of padding,

- None
- Round
- Additional

DateTime - None

When the [RangePadding](#) is set to **None**, minimum and maximum of an axis is based on the data.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    )
    .Render())
```

NONE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2017, 12, 20), y= 21 },
        new ChartData { x= new DateTime(2017, 12, 21), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 22), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 26), y= 70 },
        new ChartData { x= new DateTime(2017, 12, 27), y= 75 },
        new ChartData { x= new DateTime(2018, 01, 02), y= 82 },
        new ChartData { x= new DateTime(2018, 01, 03), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 04), y= 54 },
        new ChartData { x= new DateTime(2018, 01, 05), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 08), y= 45 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}
```

DateTime - Round

When the [RangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value, which is divisible by interval. For example, when the minimum is **15th Jan**, interval is **1** and interval type is **Month**, then the axis minimum will be **Jan 1st**.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
```

```

        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Round)
    )
    .Render()

```

ROUND.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2017, 12, 20), y= 21 },
        new ChartData { x= new DateTime(2017, 12, 21), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 22), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 26), y= 70 },
        new ChartData { x= new DateTime(2017, 12, 27), y= 75 },
        new ChartData { x= new DateTime(2018, 01, 02), y= 82 },
        new ChartData { x= new DateTime(2018, 01, 03), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 04), y= 54 },
        new ChartData { x= new DateTime(2018, 01, 05), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 08), y= 45 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}

```

DateTime - Additional

When the [RangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
        .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.Additional)
    )

```

```
)
.Render()
```

ADDITIONAL.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2017, 12, 20), y= 21 },
        new ChartData { x= new DateTime(2017, 12, 21), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 22), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 26), y= 70 },
        new ChartData { x= new DateTime(2017, 12, 27), y= 75 },
        new ChartData { x= new DateTime(2018, 01, 02), y= 82 },
        new ChartData { x= new DateTime(2018, 01, 03), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 04), y= 54 },
        new ChartData { x= new DateTime(2018, 01, 05), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 08), y= 45 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public DateTime x;
    public double y;
}
```

Label format

The date can be formatted and parsed to all globalize format using the [LabelFormat](#) property in an axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .LabelFormat("yMd")
)
.Render())
```

LABEL-FORMAT.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
```

```

{
    new ChartData { x= new DateTime(2017, 12, 20), y= 21 },
    new ChartData { x= new DateTime(2017, 12, 21), y= 24 },
    new ChartData { x= new DateTime(2017, 12, 22), y= 24 },
    new ChartData { x= new DateTime(2017, 12, 26), y= 70 },
    new ChartData { x= new DateTime(2017, 12, 27), y= 75 },
    new ChartData { x= new DateTime(2018, 01, 02), y= 82 },
    new ChartData { x= new DateTime(2018, 01, 03), y= 53 },
    new ChartData { x= new DateTime(2018, 01, 04), y= 54 },
    new ChartData { x= new DateTime(2018, 01, 05), y= 53 },
    new ChartData { x= new DateTime(2018, 01, 08), y= 45 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ChartData
{
    public DateTime x;
    public double y;
}

```

The following table describes the result of applying some common date time formats to the `LabelFormat` property.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
<code>new Date(2000, 03, 10)</code>	EEEE	Monday	The Date is displayed in day format.
<code>new Date(2000, 03, 10)</code>	yMd	04/10/2000	The Date is displayed in month/date/year format.
<code>new Date(2000, 03, 10)</code>	MMM	Apr	The Shorthand month for the date is displayed.
<code>new Date(2000, 03, 10)</code>	hm	12:00 AM	Time of the date value is displayed as label.
<code>new Date(2000, 03, 10)</code>	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format.

<!-- markdownlint-disable MD033 -->

Custom label format

Axis also supports custom label format using placeholder like `{value}°C`, in which the value represent the axis label e.g 20°C.

CSHTML

```

@ (Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>

```



```

{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
)
.PrimaryYAxis(py =>
    py.LabelFormat("${value}K")
)
.Render()

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(2017, 12, 20), y= 21 },
        new ChartData { x= new DateTime(2017, 12, 21), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 22), y= 24 },
        new ChartData { x= new DateTime(2017, 12, 26), y= 70 },
        new ChartData { x= new DateTime(2017, 12, 27), y= 75 },
        new ChartData { x= new DateTime(2018, 01, 02), y= 82 },
        new ChartData { x= new DateTime(2018, 01, 03), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 04), y= 54 },
        new ChartData { x= new DateTime(2018, 01, 05), y= 53 },
        new ChartData { x= new DateTime(2018, 01, 08), y= 45 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public DateTime x;
    public double y;
}

```

Logarithmic axis in ASP.NET MVC 3D Chart Component

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg: 10^{-6}) and higher order of magnitude (eg: 10^6).

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").

```

```

        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    )
    .PrimaryYAxis(py =>
        py.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic)
    )
    .Render();

```

LOG.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(1995, 01, 01), y= 80 },
        new ChartData { x= new DateTime(1996, 01, 01), y= 200 },
        new ChartData { x= new DateTime(1997, 01, 01), y= 400 },
        new ChartData { x= new DateTime(1998, 01, 01), y= 600 },
        new ChartData { x= new DateTime(1999, 01, 01), y= 700 },
        new ChartData { x= new DateTime(2000, 01, 01), y= 1400 },
        new ChartData { x= new DateTime(2001, 01, 01), y= 2000 },
        new ChartData { x= new DateTime(2002, 01, 01), y= 4000 },
        new ChartData { x= new DateTime(2003, 01, 01), y= 6000 },
        new ChartData { x= new DateTime(2004, 01, 01), y= 8000 },
        new ChartData { x= new DateTime(2005, 01, 01), y= 11000 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}

```

Range

The range of an axis will be calculated automatically based on the provided data and it can also be customized by using the [Minimum](#), [Maximum](#) and [Interval](#) properties of the axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>

```

```

        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    )
    .PrimaryYAxis(py =>
py.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).Minimum(100).Maximum(10000).Interval(1000)
    )
    .Render())

```

LOG-RANGE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(1995, 01, 01), y= 80 },
        new ChartData { x= new DateTime(1996, 01, 01), y= 200 },
        new ChartData { x= new DateTime(1997, 01, 01), y= 400 },
        new ChartData { x= new DateTime(1998, 01, 01), y= 600 },
        new ChartData { x= new DateTime(1999, 01, 01), y= 700 },
        new ChartData { x= new DateTime(2000, 01, 01), y= 1400 },
        new ChartData { x= new DateTime(2001, 01, 01), y= 2000 },
        new ChartData { x= new DateTime(2002, 01, 01), y= 4000 },
        new ChartData { x= new DateTime(2003, 01, 01), y= 6000 },
        new ChartData { x= new DateTime(2004, 01, 01), y= 8000 },
        new ChartData { x= new DateTime(2005, 01, 01), y= 11000 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}

```

Logarithmic base

Logarithmic base can be customized by using the [LogBase](#) property of the axis. For example when the **LogBase** is 5, the axis values follows 5^{-2} , 5^{-1} , 5^0 , 5^1 , 5^2 etc.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        ColumnSpacing(0.1).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>

```

```

        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    )
    .PrimaryYAxis(py =>
        py.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).LogBase(2)
    )
    .Render()

```

LOG-BASE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(1995, 01, 01), y= 80 },
        new ChartData { x= new DateTime(1996, 01, 01), y= 200 },
        new ChartData { x= new DateTime(1997, 01, 01), y= 400 },
        new ChartData { x= new DateTime(1998, 01, 01), y= 600 },
        new ChartData { x= new DateTime(1999, 01, 01), y= 700 },
        new ChartData { x= new DateTime(2000, 01, 01), y= 1400 },
        new ChartData { x= new DateTime(2001, 01, 01), y= 2000 },
        new ChartData { x= new DateTime(2002, 01, 01), y= 4000 },
        new ChartData { x= new DateTime(2003, 01, 01), y= 6000 },
        new ChartData { x= new DateTime(2004, 01, 01), y= 8000 },
        new ChartData { x= new DateTime(2005, 01, 01), y= 11000 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public DateTime x;
    public double y;
}

```

Logarithmic interval

The interval of the logarithmic axis can be customized by using the [Interval](#) property in the axis. When the logarithmic base is 10 and logarithmic **Interval** is 2, then the axis labels are placed at an interval of $10^{\sup>2</sup>}$. The default value of the [Interval](#) is 1.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    )
    .PrimaryYAxis(py =>

```

```
py.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic).Interval(2)
)
.Render()
```

LOG-INTERVAL.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= new DateTime(1995, 01, 01), y= 80 },
        new ChartData { x= new DateTime(1996, 01, 01), y= 200 },
        new ChartData { x= new DateTime(1997, 01, 01), y= 400 },
        new ChartData { x= new DateTime(1998, 01, 01), y= 600 },
        new ChartData { x= new DateTime(1999, 01, 01), y= 700 },
        new ChartData { x= new DateTime(2000, 01, 01), y= 1400 },
        new ChartData { x= new DateTime(2001, 01, 01), y= 2000 },
        new ChartData { x= new DateTime(2002, 01, 01), y= 4000 },
        new ChartData { x= new DateTime(2003, 01, 01), y= 6000 },
        new ChartData { x= new DateTime(2004, 01, 01), y= 8000 },
        new ChartData { x= new DateTime(2005, 01, 01), y= 11000 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public DateTime x;
    public double y;
}
```

Axis labels in ASP.NET MVC 3D Chart Component

Axis labels are the labels that are positioned adjacent to the y-axis and beneath the x-axis. It provides descriptive information about the axis.

Smart axis labels

When the axis labels overlap with each other, [LabelIntersectAction](#) property in the axis can be used to place them smartly.

Case 1: When setting `LabelIntersectAction` as `Hide`.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
```

```
.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Hide)
).Render())
```

HIDE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "South Korea",      y= 39.4 },
        new ChartData { x= "India",             y= 61.3 },
        new ChartData { x= "Pakistan",          y= 20.4 },
        new ChartData { x= "Germany",           y= 65.1 },
        new ChartData { x= "Australia",         y= 15.8 },
        new ChartData { x= "Italy",              y= 29.2 },
        new ChartData { x= "United Kingdom",     y= 44.6 },
        new ChartData { x= "Saudi Arabia",       y= 9.7 },
        new ChartData { x= "Russia",             y= 40.8 },
        new ChartData { x= "Mexico",             y= 31 },
        new ChartData { x= "Brazil",            y= 75.9 },
        new ChartData { x= "China",              y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
}
```

Case 2: When setting **LabelIntersectAction** as **Rotate45**.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate45)
).Render())
```

ROTATE45.CS

```
public ActionResult Index()
{
```

```

List<ChartData> chartData = new List<ChartData>
{
    new ChartData { x= "South Korea",      y= 39.4 },
    new ChartData { x= "India",             y= 61.3 },
    new ChartData { x= "Pakistan",          y= 20.4 },
    new ChartData { x= "Germany",           y= 65.1 },
    new ChartData { x= "Australia",         y= 15.8 },
    new ChartData { x= "Italy",             y= 29.2 },
    new ChartData { x= "United Kingdom",    y= 44.6 },
    new ChartData { x= "Saudi Arabia",      y= 9.7 },
    new ChartData { x= "Russia",            y= 40.8 },
    new ChartData { x= "Mexico",            y= 31 },
    new ChartData { x= "Brazil",            y= 75.9 },
    new ChartData { x= "China",             y= 51.4 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ChartData
{
    public string x;
    public double y;
}

```

Case 3: When setting **LabelIntersectAction** as **Rotate90**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate90)
).Render())

```

ROTATE90.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "South Korea",      y= 39.4 },
        new ChartData { x= "India",             y= 61.3 },
        new ChartData { x= "Pakistan",          y= 20.4 },
        new ChartData { x= "Germany",           y= 65.1 },
        new ChartData { x= "Australia",         y= 15.8 },
        new ChartData { x= "Italy",             y= 29.2 },
    }
}

```

```

        new ChartData { x= "United Kingdom", y= 44.6 },
        new ChartData { x= "Saudi Arabia", y= 9.7 },
        new ChartData { x= "Russia", y= 40.8 },
        new ChartData { x= "Mexico", y= 31 },
        new ChartData { x= "Brazil", y= 75.9 },
        new ChartData { x= "China", y= 51.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double y;
}

```

Edge label placement

Labels with long text at the edges of an axis may appear partially in the 3D chart. To avoid this, use the [EdgeLabelPlacement](#) property in axis, which moves the label inside the chart area for better appearance or hides it.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
    ).Render())

```

PLACEMENT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA", gold= 50 },
        new ChartData { country= "China", gold= 40 },
        new ChartData { country= "Japan", gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France", gold= 50 },
        new ChartData { country= "Germany", gold= 40 },
        new ChartData { country= "Italy", gold= 40 },
        new ChartData { country= "Sweden", gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

```



```

}
public class ChartData
{
    public string country;
    public double gold;
}

```

Maximum labels

The labels will be rendered based on the count in the [MaximumLabels](#) property per 100 pixel. If the range (minimum, maximum, interval) and [MaximumLabels](#) are set, then the priority goes to range. If the range is not set, then the priority goes to [MaximumLabels](#) property.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        Name("Product X").
        DataSource("series1").
        Add();
    })
    .PrimaryXAxis(px =>
        px.Title("Years").MaximumLabels(1)

    .EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift).MajorGridLines(mg => mg.Width(0))
    )
    .PrimaryYAxis(py =>
        py.Title("Profit ($)")

    .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None).MajorTickLines(mg => mg.Width(0))
    ).Title("Sales History of Product X")
    .Render())

<script>
    var series1 = [];
    var point1;
    var value = 80;
    var i;
    for (i = 1; i < 50; i++) {
        if (Math.random() > .5) {
            value += Math.random();
        } else {
            value -= Math.random();
        }
        point1 = { x: i, y: value.toFixed(1) };
        series1.push(point1);
    }
</script>

```

MAXIMUMLABELS.CS

```
public ActionResult Index()
{
    return View();
}
```

Axis customization in ASP.NET MVC 3D Chart Component

Title

The title for the axis can be added by using the [Title](#) property. It helps to provide quick information to the user about the data plotted in the axis. Title style can be customized using [TitleStyle](#) property of the axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    .TitleStyle(ts => ts.Size("16px").Color("grey").FontFamily("Segoe
UI").FontWeight("bold"))
    ).Render())
```

TITLE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Title rotation

The title can be rotated from 0 to 360 degree by using the [TitleRotation](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries").TitleRotation(90)
        .TitleStyle(ts => ts.Size("16px").Color("grey").FontFamily("Segoe UI").FontWeight("bold"))
    ).Render())
```

TITLEROTATION.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA", gold= 50 },
        new ChartData { country= "China", gold= 40 },
        new ChartData { country= "Japan", gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France", gold= 50 },
        new ChartData { country= "Germany", gold= 40 },
        new ChartData { country= "Italy", gold= 40 },
        new ChartData { country= "Sweden", gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Tick lines customization

The [Width](#), [Color](#) and [Height](#) of the minor and major tick lines can be customized by using the [MajorTickLines](#) and [MinorTickLines](#) properties in the axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
```

```

{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("country").
    YName("gold").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorTickLines(mt =>
mt.Color("blue").Width(5))
).Render())

```

TICK.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Grid lines customization

The [Width](#) and [Color](#) of the minor and major grid lines can be customized by using the [MajorGridLines](#) and [MinorGridLines](#) properties in the axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("country").
    YName("gold").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>

```

```
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mt =>
mt.Color("blue").Width(1))
).Render())
```

GRID.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}
```

Multiple axis

In addition to primary X and Y axis, n number of axis can be added to the chart. Series can be associated with this axis, by mapping with axis's unique name.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("silver").
YAxisName("yAxis").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.Axes(ax => {ax.RowIndex(0).Name("yAxis").Add();})
```

```
.Render() )
```

MULTIPLE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70 },
        new ChartData { country= "China",    gold= 40, silver= 60 },
        new ChartData { country= "Japan",    gold= 70, silver= 60 },
        new ChartData { country= "Australia", gold= 60, silver= 56 },
        new ChartData { country= "France",   gold= 50, silver= 45 },
        new ChartData { country= "Germany",  gold= 40, silver= 30 },
        new ChartData { country= "Italy",    gold= 40, silver= 35 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
}
```

Inversed axis

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner, set the [IsInversed](#) property to **true**.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).IsInversed(true)
)
.PrimaryYAxis(py =>
py.IsInversed(true)
).Render())
```

INVERSED.CS

```
public ActionResult Index()
{
```

```

List<ChartData> chartData = new List<ChartData>
{
    new ChartData { country= "USA",      gold= 50 },
    new ChartData { country= "China",    gold= 40 },
    new ChartData { country= "Japan",    gold= 70 },
    new ChartData { country= "Australia", gold= 60 },
    new ChartData { country= "France",   gold= 50 },
    new ChartData { country= "Germany",  gold= 40 },
    new ChartData { country= "Italy",    gold= 40 },
    new ChartData { country= "Sweden",   gold= 30 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ChartData
{
    public string country;
    public double gold;
}

```

Opposed position

To place an axis opposite from its original position, set the [OpposedPosition](#) property to **true**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
DataSource(ViewBag.dataSource).
Add();
}))
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.PrimaryYAxis(py =>
    py.OpposedPosition(true)
).Render())

```

OPPOSED.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    }
}

```

```

    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}

```

Multiple panes in ASP.NET MVC 3D Chart Component

The chart area can be divided into multiple panes using [Rows](#) and [Columns](#).

Rows

To split the chart area vertically into number of rows, use [Rows](#) property of the 3D chart.

- The space for each row can be allocated by using the [Height](#) property. The value can be either in percentage or in pixel.
- To associate a vertical axis to a particular row, specify its index to [RowIndex](#) property of the axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        Name("Germany").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y1").
        Name("Japan").
        YAxisName("yAxis").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .Rows(rows =>
    {
        rows.Height("50%").Add();
        rows.Height("50%").Add();
    })
    .Axes(ax =>
    {
        ax.OpposedPosition(true).RowIndex(1).Minimum(24).Maximum(36).Interval(4).Name("yAxis").Title("Temperature (Celsius)").LabelFormat("{value}°C").MajorGridLines(mg =>
        mg.Width(0)).Add();
    })
    .PrimaryXAxis(px =>

```



```
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Months").Interval(1)
)
.PrimaryYAxis(py =>
    py.Minimum(0).Maximum(90).Interval(20).Title("Temperature
(Fahrenheit)").LabelFormat("{value}°F")
)
.Title("Weather Condition")
.Render();
```

ROW.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Jan", y= 15, y1= 33 },
        new ChartData { x= "Feb", y= 20, y1= 31 },
        new ChartData { x= "Mar", y= 35, y1= 30 },
        new ChartData { x= "Apr", y= 40, y1= 28 },
        new ChartData { x= "May", y= 80, y1= 29 },
        new ChartData { x= "Jun", y= 70, y1= 30 },
        new ChartData { x= "Jul", y= 65, y1= 33 },
        new ChartData { x= "Aug", y= 55, y1= 32 },
        new ChartData { x= "Sep", y= 50, y1= 34 },
        new ChartData { x= "Oct", y= 30, y1= 32 },
        new ChartData { x= "Nov", y= 35, y1= 32 },
        new ChartData { x= "Dec", y= 35, y1= 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double y;
    public double y1;
}
```

For spanning the vertical axis along multiple rows, use [Span](#) property of an axis.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y").
Name("Germany").
DataSource(ViewBag.dataSource).
Add();
series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
```

```

        YName("y1").
        Name("Japan").
        YAxisName("yAxis").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .Rows(rows =>
    {
        rows.Height("50%").Add();
        rows.Height("50%").Add();
    })
    .Axes(ax =>
    {
        ax.OpposedPosition(true).RowIndex(1).Minimum(24).Maximum(36).Interval(4).Name("yAxis").Title("Temperature (Celsius)").LabelFormat("{value}°C").MajorGridLines(mg => mg.Width(0)).Add();
    })
    .PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Months").Interval(1)
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(90).Interval(20).Title("Temperature (Fahrenheit)").LabelFormat("{value}°F").Span(2)
    )
    .Title("Weather Condition")
    .Render();

```

ROW-SPAN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Jan", y= 15, y1= 33 },
        new ChartData { x= "Feb", y= 20, y1= 31 },
        new ChartData { x= "Mar", y= 35, y1= 30 },
        new ChartData { x= "Apr", y= 40, y1= 28 },
        new ChartData { x= "May", y= 80, y1= 29 },
        new ChartData { x= "Jun", y= 70, y1= 30 },
        new ChartData { x= "Jul", y= 65, y1= 33 },
        new ChartData { x= "Aug", y= 55, y1= 32 },
        new ChartData { x= "Sep", y= 50, y1= 34 },
        new ChartData { x= "Oct", y= 30, y1= 32 },
        new ChartData { x= "Nov", y= 35, y1= 32 },
        new ChartData { x= "Dec", y= 35, y1= 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;

```

```

public double y;
public double y1;
}

```

Columns

To split the chart area horizontally into number of columns, use [Columns](#) property of the 3D chart.

- The space for each column can be allocated by using the [Width](#) property. The given width can be either in percentage or in pixel.
- To associate a horizontal axis to a particular column, specify its index to [ColumnIndex](#) property of the axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("x").
    YName("y").
    Name("Germany").
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("x").
    YName("y1").
    Name("Japan").
    XAxisName("xAxis").
    DataSource(ViewBag.dataSource).
    Add();
})
.Columns(columns =>
{
    columns.Width("50%").Add();
    columns.Width("50%").Add();
})
.Axes(ax =>
{
    ax.Name("xAxis").ColumnIndex(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .MajorGridLines(mg => mg.Width(0)).Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Interval(1)
)
.PrimaryYAxis(py =>
    py.Minimum(0).Maximum(90).Interval(10).Title("Temperature (Fahrenheit)")
    .LabelFormat("{value}°F")
)
.Title("Weather Condition")
.Render())

```

COLUMN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Jan", y= 15, y1= 33 },
        new ChartData { x= "Feb", y= 20, y1= 31 },
        new ChartData { x= "Mar", y= 35, y1= 30 },
        new ChartData { x= "Apr", y= 40, y1= 28 },
        new ChartData { x= "May", y= 80, y1= 29 },
        new ChartData { x= "Jun", y= 70, y1= 30 },
        new ChartData { x= "Jul", y= 65, y1= 33 },
        new ChartData { x= "Aug", y= 55, y1= 32 },
        new ChartData { x= "Sep", y= 50, y1= 34 },
        new ChartData { x= "Oct", y= 30, y1= 32 },
        new ChartData { x= "Nov", y= 35, y1= 32 },
        new ChartData { x= "Dec", y= 35, y1= 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public double y1;
}

```

For spanning the vertical axis along multiple column, you can use [Span](#) property of an axis.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("x").
    YName("y").
    Name("Germany").
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("x").
    YName("y1").
    Name("Japan").
    XAxisName("xAxis").
    DataSource(ViewBag.dataSource).
    Add();
})
.Columns(columns =>
{
    columns.Width("50%").Add();
    columns.Width("50%").Add();
})
.Axes(ax =>
{

```

```

ax.Name("xAxis").ColumnIndex(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category).MajorGridLines(mg => mg.Width(0)).Add();
    })
    .PrimaryXAxis(px =>

px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Interval(1).Span(2)
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(90).Interval(10).Title("Temperature
(Fahrenheit)").LabelFormat("{value}°F")
    )
    .Title("Weather Condition")
    .Render()

```

COLUMN-SPAN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Jan", y= 15, y1= 33 },
        new ChartData { x= "Feb", y= 20, y1= 31 },
        new ChartData { x= "Mar", y= 35, y1= 30 },
        new ChartData { x= "Apr", y= 40, y1= 28 },
        new ChartData { x= "May", y= 80, y1= 29 },
        new ChartData { x= "Jun", y= 70, y1= 30 },
        new ChartData { x= "Jul", y= 65, y1= 33 },
        new ChartData { x= "Aug", y= 55, y1= 32 },
        new ChartData { x= "Sep", y= 50, y1= 34 },
        new ChartData { x= "Oct", y= 30, y1= 32 },
        new ChartData { x= "Nov", y= 35, y1= 32 },
        new ChartData { x= "Dec", y= 35, y1= 31 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public double y1;
}

```

Chart Types

Column chart in ASP.NET MVC 3D Chart Component

Column chart

To render a column series, use series [Type](#) as [Column](#).

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {

```

```

        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render();

```

COLUMN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Column space and width

The [ColumnSpacing](#) and [ColumnWidth](#) properties are used to customize the space between columns.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        ColumnSpacing(0.5).
        ColumnWidth(0.75).
        DataSource(ViewBag.dataSource).
        Add();
    }
)

```

```

    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render()

```

COLUMN-SPACE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50,  silver= 70 },
        new ChartData { country= "China",    gold= 40,  silver= 60 },
        new ChartData { country= "Japan",    gold= 70,  silver= 60 },
        new ChartData { country= "Australia", gold= 60,  silver= 56 },
        new ChartData { country= "France",   gold= 50,  silver= 45 },
        new ChartData { country= "Germany",  gold= 40,  silver= 30 },
        new ChartData { country= "Italy",    gold= 40,  silver= 35 },
        new ChartData { country= "Sweden",   gold= 30,  silver= 25 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
}

```

Grouped column

The data points can be grouped in the column type charts by using the [GroupName](#) property. Data points with same group name are grouped together.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column)
        .XName("Year")
        .YName("USA_Total")
        .ColumnSpacing(0.1)
        .ColumnWidth(0.7)
        .GroupName("USA")
        .DataSource(ViewBag.dataSource)
        .Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column)
        .XName("Year")
        .YName("USA_Gold")
        .ColumnSpacing(0.1)
        .ColumnWidth(0.5)
        .GroupName("USA")
    }
)

```

```

        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("Year").
        YName("UK_Total").
        ColumnSpacing(0.1).
        ColumnWidth(0.7).
        GroupName("UK").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("Year").
        YName("UK_Gold").
        ColumnSpacing(0.1).
        ColumnWidth(0.5).
        GroupName("UK").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render()

```

GROUP-COLUMN.CS

```

public ActionResult Index()
{
    List<ColumnData> chartData = new List<ColumnData>
    {
        new ColumnData { Year = "2012", USA_Total = 104, USA_Gold = 46,
        UK_Total = 65, UK_Gold = 29 },
        new ColumnData { Year = "2016", USA_Total = 121, USA_Gold = 46,
        UK_Total = 67, UK_Gold = 27 },
        new ColumnData { Year = "2020", USA_Total = 113, USA_Gold = 39,
        UK_Total = 65, UK_Gold = 22 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnData
{
    public string Year;
    public double USA_Total;
    public double USA_Gold;
    public double UK_Total;
    public double UK_Gold;
}

```

Cylindrical column chart

To render a cylindrical column chart, set the [ColumnFacet](#) property to `Cylinder` in the chart series.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)

```



```

.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column) .
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder) .
    XName("x") .
    YName("y") .
    ColumnWidth(0.9) .
    DataSource(ViewBag.dataSource) .
    Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.Render()

```

COLUMN-CYLINDER.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Czechia",      y= 1.11 },
        new ChartData { x= "Spain",        y= 1.66 },
        new ChartData { x= "USA",          y= 1.56 },
        new ChartData { x= "Germany",      y= 3.1  },
        new ChartData { x= "Russia",       y= 1.35 },
        new ChartData { x= "Slovakia",     y= 1    },
        new ChartData { x= "South Korea",  y= 3.16 },
        new ChartData { x= "France",       y= 0.92 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
}

```

Series customization

The following properties can be used to customize the [Column](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of the [Fill](#) color.

Series customization

The following properties can be used to customize the `column` series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Opacity(0.5).
        Fill("red").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render())
```

COLUMN-SERIES.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}
```

Stacked column chart in ASP.NET MVC 3D Chart Component

Stacked column chart

To render a stacked column series, use series [Type](#) as [StackingColumn](#).

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
    })
    .Render())
```

```

        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        XName("x").
        YName("y1").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        XName("x").
        YName("y2").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).LegendSettings(leg => leg.EnableHighlight(true))
    .Render()

```

STACKEDCOLUMN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new ChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new ChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new ChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new ChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new ChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new ChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new ChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new ChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new ChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Stacking group

To group the stacked columns, the [StackingGroup](#) property can be used. The columns with same group name are stacked on top of each other.

CSHTML

```

@ (Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {

```

```

series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
XName("x").
YName("y").
StackingGroup("UKAndGermany").
DataSource(ViewBag.dataSource).
Add();
series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
XName("x").
YName("y1").
StackingGroup("UKAndGermany").
DataSource(ViewBag.dataSource).
Add();
series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
XName("x").
YName("y2").
StackingGroup("UKAndGermany").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
).LegendSettings(leg => leg.EnableHighlight(true))
.Render()

```

GROUP-STACKEDCOLUMN.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new ChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new ChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new ChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new ChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new ChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new ChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new ChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new ChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new ChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Cylindrical stacked column chart

To render a cylindrical stacked column chart, set the [ColumnFacet](#) property to `Cylinder` in the chart series.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
        XName("x").
        YName("y").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
        XName("x").
        YName("y1").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
        XName("x").
        YName("y2").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).LegendSettings(leg => leg.EnableHighlight(true))
    .Render())
```

STACKEDCOLUMN-CYLINDER.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new ChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new ChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new ChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new ChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new ChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new ChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new ChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new ChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new ChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}
```

```
}
```

Series customization

The following properties can be used to customize the [Stacked Column](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of the [Fill](#) color.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        XName("x").
        YName("y").
        Fill("green").
        Opacity(0.5).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        XName("x").
        YName("y1").
        Fill("red").
        Opacity(0.5).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingColumn).
        XName("x").
        YName("y2").
        Fill("yellow").
        Opacity(0.5).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).LegendSettings(leg => leg.EnableHighlight(true))
    .Render())
```

STACKEDCOLUMN-SERIES.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new ChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new ChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new ChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new ChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new ChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new ChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
    }
```

```

        new ChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
        new ChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new ChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

100% Stacked column chart in ASP.NET MVC 3D Chart Component

100% Stacked column chart

To render a 100% stacked column series, use series [Type](#) as [StackingColumn100](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

STACKEDCOLUMN100.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
    }
}

```

```

        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

100% Cylindrical stacked column chart

To render a 100% cylindrical stacked column chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y").
    Name("UK").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y1").
    Name("Germany").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y2").
    Name("France").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("y3").
    Name("Italy").
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
    .Title("Years")
    .LabelFormat("y")
)

```



```

.PrimaryYAxis(py =>
    py.Title("GDP (%) Per Annum")
      .LabelFormat("{value}%")
      .RangePadding(Syncfusion.EJ2.Charts.ChartRangePadding.None)
    )
.Title("Gross Domestic Product Growth")
.Render()

```

STACKEDCOLUMN100-CYLINDER.CS

```

public ActionResult Index()
{
    List<CylindricalChartData> chartData = new List<CylindricalChartData>
    {
        new CylindricalChartData { x= new DateTime(2006, 01, 01), y= 900,
y1= 190, y2= 250, y3= 150 },
        new CylindricalChartData { x= new DateTime(2007, 01, 01), y= 544,
y1= 226, y2= 145, y3= 120 },
        new CylindricalChartData { x= new DateTime(2008, 01, 01), y= 880,
y1= 194, y2= 190, y3= 115 },
        new CylindricalChartData { x= new DateTime(2009, 01, 01), y= 675,
y1= 250, y2= 220, y3= 125 },
        new CylindricalChartData { x= new DateTime(2010, 01, 01), y= 765,
y1= 222, y2= 225, y3= 132 },
        new CylindricalChartData { x= new DateTime(2011, 01, 01), y= 679,
y1= 181, y2= 135, y3= 137 },
        new CylindricalChartData { x= new DateTime(2012, 01, 01), y= 770,
y1= 128, y2= 152, y3= 110 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class CylindricalChartData
{
    public DateTime x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Series customization

The following properties can be used to customize the [100% Stacked Column](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of the [Fill](#) color.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
    XName("x").
    YName("yValue").
}

```

```

        DashArray("5,5").
        Fill("red").
        Opacity(0.7).
        StackingGroup("Asia").
        Border(br => br.Width(2).Color("black")).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingColumn100).
        XName("x").
        YName("yValue1").
        DashArray("5,5").
        Fill("blue").
        Opacity(0.7).
        StackingGroup("Asia").
        Border(br => br.Width(2).Color("black")).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
    .Render()

```

STACKEDCOLUMN-100-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Bar chart in ASP.NET MVC 3D Chart Component

Bar chart

To render a bar series, use series [Type](#) as [Bar](#).

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)

```

```

.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar) .
    XName("country") .
    YName("gold") .
    DataSource(ViewBag.dataSource) .
    Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.Render()

```

BAR.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Bar space and width

The [ColumnSpacing](#) and [ColumnWidth](#) properties are used to customize the space between bars.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar) .
        XName("country") .
        YName("gold") .
        DataSource(ViewBag.dataSource) .
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar) .
        XName("country") .
        YName("silver") .
        ColumnSpacing(0.5) .
        ColumnWidth(0.75) .
    }
    )
)

```

```

        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render();

```

BAR-SPACE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50,  silver= 70 },
        new ChartData { country= "China",    gold= 40,  silver= 60 },
        new ChartData { country= "Japan",    gold= 70,  silver= 60 },
        new ChartData { country= "Australia", gold= 60,  silver= 56 },
        new ChartData { country= "France",   gold= 50,  silver= 45 },
        new ChartData { country= "Germany",  gold= 40,  silver= 30 },
        new ChartData { country= "Italy",    gold= 40,  silver= 35 },
        new ChartData { country= "Sweden",   gold= 30,  silver= 25 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
}

```

Grouped bar

The data points can be grouped in the bar type charts by using the [GroupName](#) property. Data points with same group name are grouped together.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar).
        XName("Year").
        YName("USA_Total").
        ColumnSpacing(0.1).
        ColumnWidth(0.7).
        GroupName("USA").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar).
        XName("Year").
        YName("USA_Gold").
        ColumnSpacing(0.1).

```

```

        ColumnWidth(0.5).
        GroupName("USA").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar).
        XName("Year").
        YName("UK_Total").
        ColumnSpacing(0.1).
        ColumnWidth(0.7).
        GroupName("UK").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar).
        XName("Year").
        YName("UK_Gold").
        ColumnSpacing(0.1).
        ColumnWidth(0.5).
        GroupName("UK").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render()

```

GROUP-BAR.CS

```

public ActionResult Index()
{
    List<ColumnData> chartData = new List<ColumnData>
    {
        new ColumnData { Year = "2012", USA_Total = 104, USA_Gold = 46,
        UK_Total = 65, UK_Gold = 29 },
        new ColumnData { Year = "2016", USA_Total = 121, USA_Gold = 46,
        UK_Total = 67, UK_Gold = 27 },
        new ColumnData { Year = "2020", USA_Total = 113, USA_Gold = 39,
        UK_Total = 65, UK_Gold = 22 },
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ColumnData
{
    public string Year;
    public double USA_Total;
    public double USA_Gold;
    public double UK_Total;
    public double UK_Gold;
}

```

Cylindrical bar chart

To render a cylindrical bar chart, set the [ColumnFacet](#) property to `Cylinder` in the chart series.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar).
        ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render())
```

BAR-CYLINDER.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Series customization

The following properties can be used to customize the [Bar](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of the [Fill](#) color.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Bar).
        XName("x").
```

```

        YName("y").
        Opacity(0.5).
        Fill("blue").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Render();

```

BAR-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= 2005, y= 28 },
        new ChartData { x= 2006, y= 25 },
        new ChartData { x= 2007, y= 26 },
        new ChartData { x= 2008, y= 27 },
        new ChartData { x= 2009, y= 32 },
        new ChartData { x= 2010, y= 35 },
        new ChartData { x= 2011, y= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public double x;
    public double y;
}

```

Stacked bar chart in ASP.NET MVC 3D Chart Component

Stacked bar chart

To render a stacked bar series, use series [Type](#) as [StackingBar](#).

CSHTML

```

@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y").
        Name("America").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y1").
        Name("Canada").
        DataSource(ViewBag.dataSource).
    }
)

```

```

        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y2").
        Name("France").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(py =>
        py.Interval(20)
    )
    .LegendSettings(leg => leg.EnableHighlight(true))
    .Render()

```

STACKEDBAR.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Sochi",      y= 9,      y1= 10,  y2= 4 },
        new ChartData { x= "Rio",        y= 46,     y1= 4,   y2= 10 },
        new ChartData { x= "Pyeongchang", y= 9,      y1= 11,  y2= 5 },
        new ChartData { x= "Tokyo",      y= 39,     y1= 7,   y2= 10 },
        new ChartData { x= "Beijing",    y= 8,      y1= 4,   y2= 5 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
}

```

Stacking group

To group the stacked bar, the [StackingGroup](#) property can be used. The bars with same group name are stacked on top of each other.

CSHTML

```

@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y").
        Name("America").
        StackingGroup("JohnAndAndrew").
    }
    )

```



```

        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y1").
        Name("Canada").
        StackingGroup("JohnAndAndrew").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y2").
        Name("France").
        StackingGroup("ThomasAndMichael").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(py =>
        py.Interval(20)
    )
    .LegendSettings(leg => leg.EnableHighlight(true))
    .Render();

```

GROUP-STACKEDBAR.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Sochi",          y= 9,      y1= 10,   y2= 4 },
        new ChartData { x= "Rio",            y= 46,     y1= 4,    y2= 10 },
        new ChartData { x= "Pyeongchang",    y= 9,      y1= 11,   y2= 5 },
        new ChartData { x= "Tokyo",          y= 39,     y1= 7,    y2= 10 },
        new ChartData { x= "Beijing",        y= 8,      y1= 4,    y2= 5 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
}

```

Cylindrical stacked bar chart

To render a cylindrical stacked bar chart, set the [ColumnFacet](#) property to `Cylinder` in the chart series.

CSHTML

```

@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100))

```

```

        .Series(series =>
        {
            series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
            ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
            XName("x").
            YName("y").
            Name("America").
            DataSource(ViewBag.dataSource).
            Add();
            series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
            ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
            XName("x").
            YName("y1").
            Name("Canada").
            DataSource(ViewBag.dataSource).
            Add();
            series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
            ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
            XName("x").
            YName("y2").
            Name("France").
            DataSource(ViewBag.dataSource).
            Add();
        })
        .PrimaryXAxis(px =>
            px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        )
        .PrimaryYAxis(py =>
            py.Interval(20)
        )
        .LegendSettings(leg => leg.EnableHighlight(true))
        .Render()
    
```

STACKEDBAR-CYLINDER.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Sochi",          y= 9,      y1= 10,   y2= 4  },
        new ChartData { x= "Rio",            y= 46,     y1= 4,    y2= 10 },
        new ChartData { x= "Pyeongchang",    y= 9,      y1= 11,   y2= 5  },
        new ChartData { x= "Tokyo",          y= 39,     y1= 7,    y2= 10 },
        new ChartData { x= "Beijing",        y= 8,      y1= 4,    y2= 5  }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
}
    
```

Series customization

The following properties can be used to customize the [Stacked Bar](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of the [Fill](#) color.

CSHTML

```
@(Html.EJS().Chart3D("container").EnableRotation(true).Rotation(22).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y").
        Name("America").
        Fill("red").
        Opacity(0.7).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y1").
        Name("Canada").
        Fill("green").
        Opacity(0.7).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.StackingBar).
        XName("x").
        YName("y2").
        Name("France").
        Fill("yellow").
        Opacity(0.7).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(py =>
        py.Interval(20)
    )
    .LegendSettings(leg => leg.EnableHighlight(true))
    .Render())
```

STACKEDBAR-SERIES.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Sochi",          y= 9,      y1= 10,   y2= 4 },
        new ChartData { x= "Rio",            y= 46,     y1= 4,    y2= 10 },
        new ChartData { x= "Pyeongchang",    y= 9,      y1= 11,   y2= 5 },
    }
```

```

        new ChartData { x= "Tokyo",      y= 39,    y1= 7,    y2= 10 },
        new ChartData { x= "Beijing",    y= 8,     y1= 4,    y2= 5  }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
}

```

100% Stacked bar chart in ASP.NET MVC 3D Chart Component

100% Stacked bar chart

To render a 100% stacked bar series, use series [Type](#) as [StackingBar100](#).

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue").
    DataSource(ViewBag.dataSource).
    Name("Gold").
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue1").
    DataSource(ViewBag.dataSource).
    Name("Silver").
    Add();
})
.PrimaryXAxis(px =>
    px.Interval(1)
    .ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    .IsIndexed(true)
)
.Title("Olympic Medal Counts - RIO").Render()

```

STACKEDBAR100.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
    }
}

```

```

        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

100% Cylindrical stacked bar chart

To render a cylindrical 100% stacked bar chart, set the [ColumnFacet](#) property to **Cylinder** in the chart series.

CSHTML

```

@(Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("y").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("y1").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("y2").
    DataSource(ViewBag.dataSource).
    ColumnFacet(Syncfusion.EJ2.Charts.ShapeType.Cylinder).
    Add();
}))
.Render()

```

STACKEDBAR100-CYLINDER.CS

```

public ActionResult Index()
{
    List<CylinderChartData> chartData = new List<CylinderChartData>
    {
        new CylinderChartData { x= 2000, y= 0.61, y1= 0.03, y2= 0.48 },
        new CylinderChartData { x= 2001, y= 0.81, y1= 0.05, y2= 0.53 },
        new CylinderChartData { x= 2002, y= 0.91, y1= 0.06, y2= 0.57 },
        new CylinderChartData { x= 2003, y= 1, y1= 0.09, y2= 0.61 },
        new CylinderChartData { x= 2004, y= 1.19, y1= 0.14, y2= 0.63 },
        new CylinderChartData { x= 2005, y= 1.47, y1= 0.20, y2= 0.64 },
        new CylinderChartData { x= 2006, y= 1.74, y1= 0.29, y2= 0.66 },
        new CylinderChartData { x= 2007, y= 1.98, y1= 0.46, y2= 0.76 },
    }
}

```

```

        new CylinderChartData { x= 2008, y= 1.99, y1= 0.64, y2= 0.77 },
        new CylinderChartData { x= 2009, y= 1.70, y1= 0.75, y2= 0.55 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class CylinderChartData
{
    public double x;
    public double y;
    public double y1;
    public double y2;
}

```

Series customization

The following properties can be used to customize the [100% Stacked Bar](#) series.

- [Fill](#) – Specifies the color of the series.
- [Opacity](#) – Specifies the opacity of the [Fill](#) color.

CSHTML

```

@Html.EJS().Chart("container").Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue").
    DashArray("5,5").
    Fill("blue").
    Opacity(0.7).
    Border(br => br.Width(2).Color("black")).
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.StackingBar100).
    XName("x").
    YName("yValue1").
    DashArray("5,5").
    Fill("green").
    Opacity(0.7).
    Border(br => br.Width(2).Color("black")).
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
.Render()

```

STACKEDBAR-100-SERIES.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "USA", yValue= 46, yValue1=56 },
    }
}

```

```

        new ChartData { x= "GBR", yValue= 27, yValue1=17 },
        new ChartData { x= "CHN", yValue= 26, yValue1=36 },
        new ChartData { x= "UK", yValue= 56, yValue1=16 },
        new ChartData { x= "AUS", yValue= 12, yValue1=46 },
        new ChartData { x= "IND", yValue= 26, yValue1=16 },
        new ChartData { x= "DEN", yValue= 26, yValue1=12 },
        new ChartData { x= "MEX", yValue= 34, yValue1=32},
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double yValue;
    public double yValue1;
}

```

Data labels in ASP.NET MVC 3D Chart Component

Data labels are fields that includes information about the sample point connected to an output. It can be added to a chart series by enabling the [Visible](#) property in the [DataLabel](#). By default, the labels will arrange smartly without overlapping.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataLabel(dl => dl.Visible(true)).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render())

```

DATALABEL.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA", gold= 50 },
        new ChartData { country= "China", gold= 40 },
        new ChartData { country= "Japan", gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France", gold= 50 },
        new ChartData { country= "Germany", gold= 40 },
        new ChartData { country= "Italy", gold= 40 },
    }
}

```

```

        new ChartData { country= "Sweden",    gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}

```

Position

The [Position](#) property is used to place the label either on **Top**, **Middle**, or **Bottom**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataLabel(dl =>
        dl.Visible(true).Position(Syncfusion.EJ2.Charts.Chart3DDataLabelPosition.Middle)).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render())

```

POSITION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",    gold= 50 },
        new ChartData { country= "China",  gold= 40 },
        new ChartData { country= "Japan",  gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",  gold= 50 },
        new ChartData { country= "Germany", gold= 40 },
        new ChartData { country= "Italy",   gold= 40 },
        new ChartData { country= "Sweden",  gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{

```



```

public string country;
public double gold;
}

```

Template

Label content can be formatted by using the template option. Inside the template, the placeholder text `${point.x}` and `${point.y}` can be added to display corresponding data points x & y value. Using [Template](#) property, the data label template can be set.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("country").
    YName("gold").
    DataLabel(dl => dl.Visible(true).Template("<div style=\"border: 1px
solid black; padding: 3px 3px 3px
3px\"><div>${point.x}</div><div>${point.y}</div></div>")) .
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
)
.Title("Olympic Medals")
.Render())

```

TEMPLATE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Text mapping

Text from the data source can be mapped using the [Name](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        DataLabel(dl => dl.Visible(true).Name("text")).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render())
```

MAPPING.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Jan", y= 12, text= "January : 12°C" },
        new ChartData { x= "Feb", y= 8, text= "February : 8°C" },
        new ChartData { x= "Mar", y= 11, text= "March : 11°C" },
        new ChartData { x= "Apr", y= 6, text= "April : 6°C" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public string text;
}
```

Format

Data label for the chart can be formatted using the [Format](#) property. The global formatting options can be used as 'n', 'p', and 'c'.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
```

```

        DataLabel(dl => dl.Visible(true).Format("n2")).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render();

```

FORMAT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Margin

The [Margin](#) for data label can be applied by using [Left](#), [Right](#), [Bottom](#) and [Top](#) properties.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataLabel(dl => dl.Visible(true).Border(br =>
br.Width(1).Color("red"))
        .Margin(mg => mg.Top(5).Left(5).Right(5).Bottom(5))).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render())
```

MARGIN.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Customization

The **Stroke** and **Border** of data label can be customized using [Fill](#) and [Border](#) properties.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataLabel(dl => dl.Visible(true).Border(br =>
br.Width(2).Color("red"))).
        DataSource(ViewBag.dataSource).
        Add();
    })
```

```

    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render()

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Customizing specific label

A specific label can be customized by using the [TextRender](#) event. The `TextRender` event allows you to change the label text for the point.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataLabel(dl => dl.Visible(true)).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .TextRender("textRender")
    .Render())

<script>
function textRender(args) {
    if (args.point.index === 2) {
        args.text = 'Label';
    }
}

```

```

        else {
            args.cancel = true;
        }
    }
</script>

```

SERIES-PERCENTAGE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",   gold= 46 },
        new ChartData { country= "GBR",   gold= 27 },
        new ChartData { country= "CHN",   gold= 26 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Legend in ASP.NET MVC 3D Chart Component

Legend provides information about the series rendered in the 3D chart.

Position and alignment

By using the [Position](#) property, the legend can be positioned at left, right, top or bottom of the 3D chart. The legend is positioned at the bottom of the 3D chart, by default.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
Name("Gold").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("silver").
Name("Silver").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("bronze").
Name("Bronze").
DataSource(ViewBag.dataSource).

```

```

        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg =>
lg.Visible(true).Position(Syncfusion.EJ2.Charts.LegendPosition.Top)
    )
    .Render();

```

POSITION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

The custom position helps you to position the legend anywhere in the 3D chart using X and Y coordinates.

CSHTML

```

@ (Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>

```

```

{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("country").
    YName("gold").
    Name("Gold").
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("country").
    YName("silver").
    Name("Silver").
    DataSource(ViewBag.dataSource).
    Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
    XName("country").
    YName("bronze").
    Name("Bronze").
    DataSource(ViewBag.dataSource).
    Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
)
.PrimaryYAxis(py =>
    py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
)
.Title("Olympic Medals")
.LegendSettings(lg => lg.Visible(true)
    .Position(Syncfusion.EJ2.Charts.LegendPosition.Custom)
    .Location(lc => lc.X(200).Y(20))
.Render())

```

CUSTOM-POSITION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",          gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",        gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",         gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia",     gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",        gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",       gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",         gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",        gold= 30, silver= 25, bronze=
27 }
    };
}

```



```

        ViewBag.dataSource = chartData;
        return View();
    }
    public class ChartData
    {
        public string country;
        public double gold;
        public double silver;
        public double bronze;
    }

```

Legend reverse

The order of the legend items can be reversed by using the [Reverse](#) property. By default, legend for the first series in the collection will be placed first.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg => lg.Visible(true).Reverse(true))
    .Render())

```

REVERSE.CS

```

public ActionResult Index()
{

```

```

List<ChartData> chartData = new List<ChartData>
{
    new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
    new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
    new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
    new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
    new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
    new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
    new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
    new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend alignment

The legend can be aligned at near, far or center to the 3D chart using the [Alignment](#) property.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
Name("Gold").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("silver").
Name("Silver").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("bronze").
Name("Bronze").
DataSource(ViewBag.dataSource).
Add();
}
)

```

```

        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg =>
lg.Visible(true).Position(Syncfusion.EJ2.Charts.LegendPosition.Top)
        .Alignment(Syncfusion.EJ2.Charts.Alignment.Near))
    .Render())

```

ALIGNMENT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",          gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",        gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",        gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia",    gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",       gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",     gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",       gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",      gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend customization

To change the legend icon shape, [LegendShape](#) property in the [Series](#) can be used. By default, the legend icon shape is `SeriesType`.

CSHTML

```

@ (Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.SeriesType).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Rectangle).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg => lg.Visible(true))
    .Render())

```

CUSTOM.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",          gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",        gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",        gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia",    gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",       gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",     gold= 40, silver= 30, bronze=
22 },
    }
}

```

```

        new ChartData { country= "Italy",      gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",    gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend size

By default, legend takes 20% - 25% of the 3D chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the 3D chart's width vertically, when it is placed on left or right position. You can change this default legend size by using the [Height](#) and [Width](#) properties of the [LegendSettings](#).

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
py.Minimum(0).Maximum(80).Interval(20).Title("Medals")

```

```

    )
    .Title("Olympic Medals")
    .LegendSettings(lg =>
lg.Visible(true).Width("500").Height("100").Border(br =>
br.Width(1).Color("pink")))
    .Render())

```

SIZE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend item size

The size of the legend items can be customised by using the [ShapeHeight](#) and [ShapeWidth](#) properties.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
Name("Gold").
LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
DataSource(ViewBag.dataSource).

```

```

        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        LegendShape(Syncfusion.EJ2.Charts.LegendShape.Circle).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg => lg.Visible(true).ShapeHeight(10).ShapeWidth(10))
    .Render();

```

ITEM-SIZE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;

```

```

public double gold;
public double silver;
public double bronze;
}

```

Paging for legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. Each legend items can be viewed by navigating between the pages using navigation buttons.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y").
Name("December 2007").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y1").
Name("December 2008").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y2").
Name("December 2009").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("x").
YName("y3").
Name("December 2010").
DataSource(ViewBag.dataSource).
Add();
})
.PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries").Interval(1)

.LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate45)
)
.PrimaryYAxis(py =>
py.Minimum(0).Maximum(90).LabelFormat("{value}%").Title("Penetration (%)")
)
.Title("FB Penetration of Internet Audience")
.LegendSettings(lg =>
lg.Visible(true).Padding(10).ShapePadding(10).Width("200").Border(br =>
br.Width(2).Color("grey")))
.Render())

```


PAGING.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "WW",      y= 12,    y1= 22,    y2= 38.3, y3= 50
    },
        new ChartData { x= "EU",      y= 9.9,    y1= 26,    y2= 45.2, y3= 63.6
    },
        new ChartData { x= "APAC",    y= 4.4,    y1= 9.3,    y2= 18.2, y3= 20.9
    },
        new ChartData { x= "LATAM",   y= 6.4,    y1= 28,    y2= 46.7, y3= 65.1
    },
        new ChartData { x= "MEA",     y= 30,     y1= 45.7, y2= 61.5, y3= 73
    },
        new ChartData { x= "NA",      y= 25.3, y1= 35.9, y2= 64,    y3= 81.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Legend text wrap

When the legend text exceeds the container, the text can be wrapped by using the [TextWrap](#) property. End user can also wrap the legend text based on the [MaximumLabelWidth](#) property.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").

```

```

        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg =>
lg.Visible(true).Position(Syncfusion.EJ2.Charts.LegendPosition.Right).TextWr
ap(Syncfusion.EJ2.Charts.TextWrap.Wrap).MaximumLabelWidth(50))
    .Render())

```

TEXT-WRAP.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Series selection through legend

By default, you can collapse the series visibility by clicking the legend. On the other hand, turn off the [ToggleVisibility](#) property if you must use a legend click to choose a series.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg => lg.Visible(true).ToggleVisibility(false))
    .Render())

```

SELECTION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",          gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",        gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",        gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia",    gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",       gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",     gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",       gold= 40, silver= 35, bronze=
37 },
    }
}

```

```

27     new ChartData { country= "Sweden",    gold= 30, silver= 25, bronze=
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Collapsing legend item

By default, series name will be displayed as legend. To skip the legend for a particular series, you can give empty string to the series name.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries")
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg => lg.Visible(true).ToggleVisibility(true))
    .Render())

```

COLLAPSE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Legend title

You can set title for legend using [Title](#) property in [LegendSettings](#). The [Size](#), [Color](#), [Opacity](#), [FontStyle](#), [FontWeight](#), [FontFamily](#), [TextAlignment](#), and [TextOverflow](#) of legend title can be customized by using the [TitleStyle](#) property in [LegendSettings](#). The [TitlePosition](#) is used to set the legend position in **Top**, **Left** and **Right** position. The [MaximumTitleWidth](#) is used to set the width of the legend title. By default, it will be 100px.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
    })

```

```

        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(80).Interval(20).Title("Medals")
    )
    .Title("Olympic Medals")
    .LegendSettings(lg => lg.Visible(true).Title("Countries"))
    .Render();

```

TITLE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Arrow page navigation

The page number will always be visible while using legend paging. It is now possible to disable the page number and enable page navigation with the left and right arrows. The [EnablePages](#) property needs to be set to **false** in order to render the arrow page navigation.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        Name("December 2007").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y1").
        Name("December 2008").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y2").
        Name("December 2009").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y3").
        Name("December 2010").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries").Interval(1)

    .LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate45)
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(90).LabelFormat("{value}%").Title("Penetration (%)")
    )
    .Title("FB Penetration of Internet Audience")
    .LegendSettings(lg => lg.Width("180").EnablePages(false))
    .Render())
```

ARROW-PAGE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
```

```

    {
        new ChartData { x= "WW",      y= 12,    y1= 22,    y2= 38.3, y3= 50
    },
        new ChartData { x= "EU",      y= 9.9,    y1= 26,    y2= 45.2, y3= 63.6
    },
        new ChartData { x= "APAC",    y= 4.4,    y1= 9.3,    y2= 18.2, y3= 20.9
    },
        new ChartData { x= "LATAM",   y= 6.4,    y1= 28,    y2= 46.7, y3= 65.1
    },
        new ChartData { x= "MEA",     y= 30,     y1= 45.7, y2= 61.5, y3= 73
    },
        new ChartData { x= "NA",      y= 25.3, y1= 35.9, y2= 64,    y3= 81.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Legend item padding

The [ItemPadding](#) property can be used to adjust the space between the legend items.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        Name("December 2007").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y1").
        Name("December 2008").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y2").
        Name("December 2009").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y3").
        Name("December 2010").
    }
)

```



```

        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category).Title("Countries").Interval(1)

    .LabelIntersectAction(Syncfusion.EJ2.Charts.LabelIntersectAction.Rotate45)
    )
    .PrimaryYAxis(py =>
        py.Minimum(0).Maximum(90).LabelFormat("{value}%").Title("Penetration (%)")
    )
    .Title("FB Penetration of Internet Audience")
    .LegendSettings(lg => lg.ItemPadding(30).EnablePages(false))
    .Render()

```

ITEM-PADDING.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "WW",      y= 12,      y1= 22,      y2= 38.3, y3= 50
    },
        new ChartData { x= "EU",      y= 9.9,      y1= 26,      y2= 45.2, y3= 63.6
    },
        new ChartData { x= "APAC",    y= 4.4,      y1= 9.3,      y2= 18.2, y3= 20.9
    },
        new ChartData { x= "LATAM",   y= 6.4,      y1= 28,      y2= 46.7, y3= 65.1
    },
        new ChartData { x= "MEA",     y= 30,      y1= 45.7,    y2= 61.5, y3= 73
    },
        new ChartData { x= "NA",      y= 25.3,    y1= 35.9,    y2= 64,   y3= 81.4 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public double y1;
    public double y2;
    public double y3;
}

```

Tooltip in ASP.NET MVC 3D Chart Component

The 3D Chart will display details about the points through tooltip, when the mouse is moved over the specific point.

Default tooltip

By default, tooltip is not visible. The tooltip can be enabled by setting the [Enable](#) property in `TooltipSettings` to **true**.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    )
    .Tooltip(tp => tp.Enable(true))
    .Render())
```

DEFAULT-TOOLTIP.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}
```

Fixed tooltip

By default, tooltip track the mouse movement, but the tooltip can be set in fixed position by using the [Location](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    )
    .Tooltip(tp => tp.Enable(true).Location(lc => lc.X(120).Y(20)))
    .Render())
```

FIXED-TOOLTIP.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}
```

Format the tooltip

By default, tooltip shows information of x and y value in points. In addition to that, more information can be shown in tooltip by using the [Format](#) property. For example the format `${series.name} : ${point.y}` shows series name and point y value.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        Name("Month").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    )
    .Tooltip(tp =>
tp.Enable(true).Header("Unemployment").Format("<b>${series.name} : 
${point.y}</b>")
    ).Render())
```

FORMAT-TOOLTIP.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}
```

Tooltip template

Any HTML elements can be displayed in the tooltip by using the [Template](#) property of the tooltip. The `#{x}` and `#{y}` can be used as place holders in the HTML element to display the x and y values of the corresponding data point.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    )
    .Tooltip(tp => tp.Enable(true).Template("#Unemployment"))
    .Render())
```

TEMPLATE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}
```

Customize the appearance of tooltip

The [Fill](#) and [Border](#) properties are used to customize the background color and border of the tooltip respectively. The [TextStyle](#) property in the tooltip is used to customize the font of the tooltip text.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("month").
        YName("sales").
        Name("China").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

    .LabelPlacement(Syncfusion.EJ2.Charts.LabelPlacement.BetweenTicks).LabelRotation(-45)
    )
    .Tooltip(tp => tp.Enable(true).Format("${point.x} :
    ${point.y}").Fill("#7bb4eb")
        .Border(br => br.Width(2).Color("grey")))
    .Render())
```

CUSTOM-TOOLTIP.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { month= "Jan", sales= 35 },
        new ChartData { month= "Feb", sales= 28 },
        new ChartData { month= "Mar", sales= 34 },
        new ChartData { month= "Apr", sales= 32 },
        new ChartData { month= "May", sales= 40 },
        new ChartData { month= "Jun", sales= 32 },
        new ChartData { month= "Jul", sales= 35 },
        new ChartData { month= "Aug", sales= 55 },
        new ChartData { month= "Sep", sales= 38 },
        new ChartData { month= "Oct", sales= 30 },
        new ChartData { month= "Nov", sales= 25 },
        new ChartData { month= "Dec", sales= 32 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string month;
    public double sales;
}
```

Selection in ASP.NET MVC 3D Chart Component

The 3D chart provides selection support for the series and its data points on mouse click.

When mouse is clicked on the data points, the corresponding series legend will also be selected.

We have different types of selection mode for selecting a data.

- None
- Point
- Series
- Cluster

Point

To select a point, set the [SelectionMode](#) property to **Point**.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .SelectionMode(Syncfusion.EJ2.Charts.Chart3DSelectionMode.Point)
    .Render())
```

POINT-SELECTION.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",          gold= 50, silver= 70, bronze=
45 },
```

```

55     new ChartData { country= "China",      gold= 40, silver= 60, bronze=
50     },
    new ChartData { country= "Japan",      gold= 70, silver= 60, bronze=
40     },
    new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
35     },
    new ChartData { country= "France",     gold= 50, silver= 45, bronze=
22     },
    new ChartData { country= "Germany",    gold= 40, silver= 30, bronze=
37     },
    new ChartData { country= "Italy",      gold= 40, silver= 35, bronze=
27     },
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Series

To select a series, set the [SelectionMode](#) property to **Series**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
Name("Gold").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("silver").
Name("Silver").
DataSource(ViewBag.dataSource).
Add();
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("bronze").
Name("Bronze").
DataSource(ViewBag.dataSource).
Add();
}))
.PrimaryXAxis(px =>
    px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)

```



```

    )
    .Title("Olympic Medals")
    .SelectionMode(Syncfusion.EJ2.Charts.Chart3DSelectionMode.Series)
    .Render()

```

SERIES-SELECTION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Cluster

To select the points that corresponds to the same index in all the series, set the [SelectionMode](#) property to **Cluster**.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
.Rotation(7).Tilt(10).Depth(100)
.Series(series =>
{
    series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
XName("country").
YName("gold").
Name("Gold").
DataSource(ViewBag.dataSource).
Add();
}
)

```

```

        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .SelectionMode(Syncfusion.EJ2.Charts.Chart3DSelectionMode.Cluster)
    .Render()

```

CLUSTER-SELECTION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Selection type

To select multiple points or series, enable the [IsMultiSelect](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .SelectionMode(Syncfusion.EJ2.Charts.Chart3DSelectionMode.Series)
    .IsMultiSelect(true)
    .Render())
```

SELECTION-TYPE.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",          gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",        gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",        gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia",    gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",       gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",     gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",        gold= 40, silver= 35, bronze=
37 },
    }
```

```

27     new ChartData { country= "Sweden",    gold= 30, silver= 25, bronze=
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Selection during initial loading

In a 3D chart, selecting a point or series during initial loading can only be done programmatically. The [SelectedDataIndexes](#) property can be used for this.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .SelectionMode(Syncfusion.EJ2.Charts.Chart3DSelectionMode.Point)
    .IsMultiSelect(true)
    .SelectedDataIndexes(ViewBag.selectedData)
    .Render())

```

ONLOAD.CS

```

public ActionResult Index()

```

```

{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45    },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55    },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50    },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40    },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35    },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22    },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37    },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27    }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Selection through legend

To select a point or series through on legend use [ToggleVisibility](#) property. Also, use [EnableHighlight](#) property for highlighting the series through legend.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").

```

```

        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .LegendSettings(lg =>
        lg.Visible(true).ToggleVisibility(false).EnableHighlight(true))
    .Render();

```

SELECTION-LEGEND.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Print and Export in ASP.NET MVC 3D Chart Component

Print

The rendered 3D chart can be printed directly from the browser by calling the public method **print**. The ID of the 3D chart's div element must be passed as the input parameter to that method.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)

```

```

        .Series(series =>
        {
            series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column) .
            XName("country") .
            YName("gold") .
            Name("Gold") .
            DataSource(ViewBag.dataSource) .
            Add();
        })
        .PrimaryXAxis(px =>
            px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
        )
        .Title("Olympic Medals")
        .Render()
</div>

@Html.EJS().Button("togglebtn").IsPrimary(true).Content("Print").Render()
</div>
<script>
    document.getElementById('togglebtn').onclick = () => {
        var chart = document.getElementById('container').ej2_instances[0];
        chart.print();
    };
</script>

```

PRINT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Export

The rendered 3D chart can be exported to JPEG, PNG, SVG, or PDF format using the `export` method. The input parameters for this method are `type` for format and `fileName` for result.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true)
    .Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    )
    .Title("Olympic Medals")
    .Render())
<div>
    @(Html.EJS().Button("export").IsPrimary(true).Content("Export").Render())
</div>
<script>
    document.getElementById('export').onclick = () => {
        var chart = document.getElementById('container').ej2_instances[0];
        chart.export('PDF', 'result');
    };
</script>

```

EXPORT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Appearance in ASP.NET MVC 3D Chart Component

Custom color palette

The default color of series or points can be customized by providing a custom color palette of your choice by using the [Palettes](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        Name("Gold").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("silver").
        Name("Silver").
        DataSource(ViewBag.dataSource).
        Add();
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("bronze").
        Name("Bronze").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Palettes(new string[] { "#E94649", "#F6B53F", "#6FAAB0", "#C4C24A" })
    .Render())
```

CUSTOM.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50, silver= 70, bronze=
45 },
        new ChartData { country= "China",    gold= 40, silver= 60, bronze=
55 },
        new ChartData { country= "Japan",    gold= 70, silver= 60, bronze=
50 },
        new ChartData { country= "Australia", gold= 60, silver= 56, bronze=
40 },
        new ChartData { country= "France",   gold= 50, silver= 45, bronze=
35 },
        new ChartData { country= "Germany",  gold= 40, silver= 30, bronze=
22 },
        new ChartData { country= "Italy",    gold= 40, silver= 35, bronze=
37 },
        new ChartData { country= "Sweden",   gold= 30, silver= 25, bronze=
27 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
```

```

}
public class ChartData
{
    public string country;
    public double gold;
    public double silver;
    public double bronze;
}

```

Data point customization

The color of an individual data point can be customized using the below options.

Point color mapping

The color for the points can be bound from the [DataSource](#) for the series by utilizing the [PointColorMapping](#) property.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("x").
        YName("y").
        PointColorMapping("color").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render())

```

POINT-COLOR.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { x= "Jan", y= 6.96, color= "#ed4c40" },
        new ChartData { x= "Feb", y= 8.9, color= "#3285f3" },
        new ChartData { x= "Mar", y= 12, color= "#1dd7f3" },
        new ChartData { x= "Apr", y= 17.5, color= "#fe1684" },
        new ChartData { x= "May", y= 22.1, color= "#4633f2" }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string x;
    public double y;
    public string color;
}

```

Point level customization

The data label and fill color of each data point can be customized using the [PointRender](#) and [TextRender](#) events.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .PointRender("pointRender")
    .Render())

<script>
var colors = ["#00bdae", "#404041", "#357cd2", "#e56590", "#f8b883",
    "#70ad47", "#dd8abd", "#7f84e8", "#7bb4eb", "#ea7a57"];
function pointRender(args) {
    args.fill = colors[args.point.index];
}
</script>
```

POINT-CUSTOM.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA", gold= 50 },
        new ChartData { country= "China", gold= 40 },
        new ChartData { country= "Japan", gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France", gold= 50 },
        new ChartData { country= "Germany", gold= 40 },
        new ChartData { country= "Italy", gold= 40 },
        new ChartData { country= "Sweden", gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

<!-- markdownlint-disable MD036 -->

Chart area customization

<!-- markdownlint-disable MD036 -->

Customize the chart background

The background color and border of the 3D chart can be customized using the [Background](#) and [Border](#) properties.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Background("skyblue")
    .Border(br => br.Color("#FF0000").Width(2))
    .Render())
```

BACKGROUND.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Chart margin

The 3D chart's margin can be set from its container using the [Margin](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Background("skyblue")
    .Border(br => br.Color("#FF0000").Width(2))
    .Margin(mr => mr.Left(40).Right(40).Top(40).Bottom(40))
    .Render())
```

MARGIN.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Animation

To customize the animation for a particular series, the [Animation](#) property can be used. It can be enabled or disabled by using the [Enable](#) property. The [Duration](#) property specifies the duration of an animation and the [Delay](#) property allows us to start the animation at desire time.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
```

```

        YName("gold").
        Animation(an => an.Enable(true).Duration(2000).Delay(200)).
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render()

```

ANIMATION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Chart rotation

The 3D Chart can be rotated by using the [EnableRotation](#) property.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render()

```

ROTATION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}

```

Title

The 3D chart can be given a title by using [Title](#) property, to show the information about the data plotted.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals")
    .Render())

```

TITLE.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    }
}

```

```

    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}

```

Title position

By using the [Position](#) property in [TitleStyle](#), the [Title](#) can be positioned at left, right, top or bottom of the 3D chart. The title is positioned at the top of the 3D chart, by default.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals").TitleStyle(ts =>
ts.Position(Syncfusion.EJ2.Charts.TitlePosition.Bottom))
    .Render())

```

TITLEPOSITION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}
public class ChartData
{
    public string country;
    public double gold;
}

```


The custom option is used to position the title anywhere in the 3D chart using [X](#) and [Y](#) coordinates.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals").TitleStyle(ts =>
ts.Position(Syncfusion.EJ2.Charts.TitlePosition.Custom).X(300).Y(60))
    .Render())
```

TITLEOPTION.CS

```
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}
```

Title alignment

The title can be aligned to the near, far, or center of the 3D chart by using the [TextAlignment](#) property.

CSHTML

```
@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {
        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
```

```

        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals").TitleStyle(ts =>
        ts.TextAlignment(Syncfusion.EJ2.Charts.Alignment.Far))
    .Render();

```

TITLEALIGNMENT.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Title customization

The [TitleStyle](#) property of the 3D chart provides options to customize the title by using the following properties.

- [Size](#) - Specifies the size of the title.
- [Color](#) - Specifies the color for the title.
- [FontFamily](#) - Specifies the font family for the title.
- [FontWeight](#) - Specifies the font weight of the title.
- [FontStyle](#) - Specifies the font style for the title.
- [Opacity](#) - Specifies the opacity for the color of the title.
- [TextAlignment](#) - Specifies the alignment of the title.
- [TextOverflow](#) - Specifies the overflow of the title.

CSHTML

```

@(Html.EJS().Chart3D("container").WallColor("transparent").EnableRotation(true).Rotation(7).Tilt(10).Depth(100)
    .Series(series =>
    {

```

```

        series.Type(Syncfusion.EJ2.Charts.Chart3DSeriesType.Column).
        XName("country").
        YName("gold").
        DataSource(ViewBag.dataSource).
        Add();
    })
    .PrimaryXAxis(px =>
        px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)
    ).Title("Olympic Medals").TitleStyle(ts =>
        ts.Size("18px").Color("red").TextOverflow(Syncfusion.EJ2.Charts.TextOverflow
        .Wrap))
    .Render();

```

CUSTOMIZATION.CS

```

public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData { country= "USA",      gold= 50 },
        new ChartData { country= "China",    gold= 40 },
        new ChartData { country= "Japan",    gold= 70 },
        new ChartData { country= "Australia", gold= 60 },
        new ChartData { country= "France",   gold= 50 },
        new ChartData { country= "Germany",  gold= 40 },
        new ChartData { country= "Italy",    gold= 40 },
        new ChartData { country= "Sweden",   gold= 30 }
    };
    ViewBag.dataSource = chartData;
    return View();
}

public class ChartData
{
    public string country;
    public double gold;
}

```

Accessibility in ASP.NET MVC 3D Chart Component

Accessibility is achieved in the 3D chart component through WAI-ARIA standard and keyboard navigation. The 3D chart features can be effectively accessed through assistive technologies such as screen readers.

WAI-ARIA

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with AJAX, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

- img (role)
- button (role)
- region (role)
- aria-label (attribute)
- aria-hidden (attribute)

- aria-pressed (attribute)

Keyboard navigation

All the 3D chart actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below.

Interaction Keys | Description

Tab | Moves the focus to the next element in the chart.

Shift + Tab | Moves the focus to the previous element in the chart.

DownArrow | Moves the focus to the data point left side from the selected point.

UpArrow | Moves the focus to the data point right side from the selected point.

Left Arrow | Moves the focus to the next series in the chart.

Right Arrow | Moves the focus to the previous series in the chart.

ESC | Cancel the tooltip for the data point.

Enter/Space | Selects the data point in the series.

Down/Left Arrow | Moves the focus to the legend left side from the selected legend.

Up/Right Arrow | Moves the focus to the legend right side from the selected legend.

Enter/Space | Toggles the visibility of the corresponding series.

Ctrl + P | Print the chart.

CheckBox

Getting Started with ASP.NET MVC CheckBox Control

This section briefly explains about how to include [ASP.NET MVC CheckBox](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

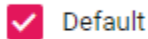
Add ASP.NET MVC CheckBox control

Now, add the Syncfusion ASP.NET MVC CheckBox control in ~/Home/Index.cshtml page.

CSHTML

```
@Html.EJS().CheckBox("default").Label("Default").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC CheckBox control will be rendered in the default web browser.



Change the CheckBox state

The Essential JS 2 CheckBox contains 3 different states visually, they are:

- Checked
- Unchecked
- Indeterminate

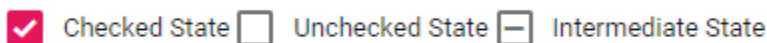
The CheckBox [Checked](#) property is used to handle the checked and unchecked state. In checked state a tick mark will be added to the visualization of CheckBox.

Indeterminate

The CheckBox indeterminate state can be set through [Indeterminate](#) property. CheckBox indeterminate state masks the real value of CheckBox visually. The Checkbox cannot be changed to indeterminate state through the user interface, this state can be achieved only through the property.

CSHTML

```
@Html.EJS().CheckBox("checked").Label("Checked State").Checked(true).Render()  
@Html.EJS().CheckBox("unchecked").Label("Unchecked State").Render()  
@Html.EJS().CheckBox("indeterminate").Label("Intermediate State").Indeterminate(true).Render()  
  
<style>  
.e-checkbox-wrapper {  
    margin-top: 18px;  
}  
li {  
    list-style: none;  
}  
</style>
```



Note: [View Sample in GitHub.](#)

Label and Size in Check Box Control

This section explains the different sizes and labels.

Label

The CheckBox caption can be defined by using the [Label](#) property. This reduces the manual addition of label for CheckBox. You can customize the label position before or after the CheckBox through the [LabelPosition](#) property.

CSHTML

```
@Html.EJS().CheckBox("cbox1").Label("Left Side  
Label").LabelPosition(Syncfusion.EJ2.Buttons.LabelPosition.Before).Render()  
@Html.EJS().CheckBox("cbox2").Label("Right Side  
Label").LabelPosition(Syncfusion.EJ2.Buttons.LabelPosition.After).Render()  
<style>  
.e-checkbox-wrapper {  
    margin-top: 18px;  
}  
li {  
    list-style: none;  
}  
</style>
```

LABEL.CS

```
public ActionResult Label()  
{  
    return View();  
}
```

Size

The different CheckBox sizes available are default and small. To reduce the size of default CheckBox to small, set the [CssClass](#) property to `e-small`.

CSHTML

```
@Html.EJS().CheckBox("cbox1").Label("Small").CssClass("e-small").Render()  
@Html.EJS().CheckBox("cbox2").Label("Default").Render()  
<style>  
.e-checkbox-wrapper {  
    margin-top: 18px;  
}  
li {  
    list-style: none;  
}  
</style>
```

SIZE.CS

```
public ActionResult Size()  
{  
    return View();  
}
```

Note: [View Sample in GitHub.](#)

See also

- [CheckBox customization](#)

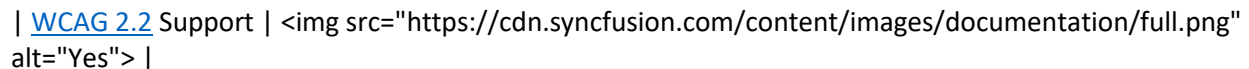
Accessibility in CheckBox Control

The Check box component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Check box component is outlined below.

| Accessibility Criteria | Compatibility |

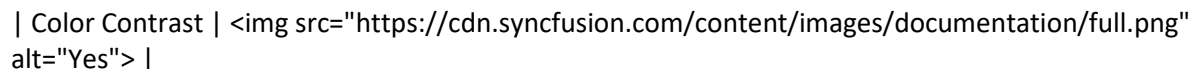
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Yes"> |

| Mobile Device Support |  alt="Yes"> |

| Keyboard Navigation Support |  alt="Yes"> |

| [Accessibility Checker](#) Validation |  alt="Yes"> |

| [Axe-core](#) Accessibility Validation |  alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> alt="Yes"> - All features of the component meet the requirement.</div>

<div> alt="Intermediate"> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Check box component followed the [WAI-ARIA](https://www.w3.org/WAI/ARIA/apg/patterns/Check box/) patterns to meet the accessibility. The following ARIA attributes are used in the Check box component:

| Attributes | Purpose |

| --- | --- |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Check box component followed the [keyboard interaction](https://www.w3.org/WAI/ARIA/apg/patterns/Check box/#keyboardinteraction) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Check box component.

| Press | To do this |

| --- | --- |

| **Space** | When the Check box has focus, pressing the Space key changes the state of the Check box. |

Ensuring accessibility

The Check box component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Check box component is shown in the following sample. Open the [sample](https://ej2.syncfusion.com/accessibility/Check box.html) in a new window to evaluate the accessibility of the Check box component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Styles and Appearances

To modify the CheckBox appearance, you need to override the default CSS of CheckBox component. Find the list of CSS classes and its corresponding section in CheckBox. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ----- | ----- |

| **.e-checkbox-wrapper .e-frame** | To customize the checkbox frame. |

| **.e-checkbox-wrapper:hover .e-frame** | To customize the checkbox frame on hover. |

| **.e-checkbox-wrapper .e-label** | To customize the checkbox label. |

| **.e-checkbox-wrapper:hover .e-label** | To customize the checkbox label on hover. |

| .e-checkbox-wrapper .e-frame.e-check | To customize the checked checkbox. |

| .e-checkbox-wrapper:hover .e-frame.e-check | To customize the checked checkbox when hover |

 Primary

 Success

 Info

 Warning

 Danger

How To

Customized CheckBox Control

Customize CheckBox Appearance

You can customize the appearance of the CheckBox component using the CSS rules. Define own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

The background and border color of the CheckBox is customized through the custom classes to create primary, success, warning, and danger info type of checkbox.

CSHTML

```
@Html.EJS().CheckBox("primary").Label("Primary").Checked(true).CssClass("e-
primary").Render()
@Html.EJS().CheckBox("success").Label("Success").Checked(true).CssClass("e-
success").Render()
@Html.EJS().CheckBox("info").Label("Info").Checked(true).CssClass("e-
info").Render()
@Html.EJS().CheckBox("warning").Label("Warning").Checked(true).CssClass("e-
warning").Render()
@Html.EJS().CheckBox("danger").Label("Danger").Checked(true).CssClass("e-
danger").Render()
<style>
.e-checkbox-wrapper {
    margin-top: 18px;
}
li {
    list-style: none;
}
.e-checkbox-wrapper.e-primary:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
    background-color: #e03872;
}
.e-checkbox-wrapper.e-success .e-frame.e-check,
.e-checkbox-wrapper.e-success .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
    background-color: #689f38;
```

```

}
.e-checkbox-wrapper.e-success:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
    background-color: #449d44;
}
.e-checkbox-wrapper.e-info .e-frame.e-check,
.e-checkbox-wrapper.e-info .e-checkbox:focus + .e-frame.e-check { /* csslint
allow: adjoining-classes */
    background-color: #2196f3;
}
.e-checkbox-wrapper.e-info:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
    background-color: #0b7dda;
}
.e-checkbox-wrapper.e-warning .e-frame.e-check,
.e-checkbox-wrapper.e-warning .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
    background-color: #ef6c00;
}
.e-checkbox-wrapper.e-warning:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
    background-color: #cc5c00;
}
.e-checkbox-wrapper.e-danger .e-frame.e-check,
.e-checkbox-wrapper.e-danger .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
    background-color: #d84315;
}
.e-checkbox-wrapper.e-danger:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
    background-color: #ba3912;
}
</style>

```

CUSTOM.CS

```

public ActionResult Custom()
{
    return View();
}

```

Customize width and height

The height and width of the CheckBox component can be customized by setting **height** and **width** property. The following section explains about how to customize the height and width of the CheckBox component.

CSHTML

```

@Html.EJS().CheckBox("customsize").Label("Default").CssClass("e-
customsize").Render()
<style>
.e-checkbox-wrapper {
    margin-top: 30px;
}
.e-customsize.e-checkbox-wrapper .e-frame {

```

```

    height: 30px;
    width: 30px;
    padding: 8px 0;
}
.e-customsize.e-checkbox-wrapper .e-check {
    font-size: 20px;
}
.e-customsize.e-checkbox-wrapper .e-ripple-container {
    height: 52px;
    top: -11px;
    width: 47px;
}
.e-customsize.e-checkbox-wrapper .e-label {
    line-height: 30px;
    font-size: 20px;
}
</style>

```

CUSTOMHEIGHT.CS

```

public ActionResult CustomHeight()
{
    return View();
}

```

Custom Frame

CheckBox frame can be customized as per the requirement by adding CSS rules.

In the following example, to-do list is displayed with round checkbox by changing **border-radius** as **100%** by adding **e-custom** class.

CSHTML

```

@Html.EJS().CheckBox("cbox2").Label("Buy
Groceries").Checked(true).CssClass("e-custom").Render()
@Html.EJS().CheckBox("cbox3").Label("Pay Rent").Checked(true).CssClass("e-
custom").Render()
@Html.EJS().CheckBox("cbox4").Label("Make Dinner").CssClass("e-
custom").Render()
@Html.EJS().CheckBox("cbox1").Label("Finish To-do List
Article").CssClass("e-custom").Render()
<style>
    li {
        list-style: none;
    }
    .e-checkbox-wrapper {
        margin-top: 18px;
    }
    e-custom .e-frame {
        border-radius: 100%;
    }
</style>

```

CUSTOMSIZE.CS

```
public ActionResult CustomSize()
{
    return View();
}
```

Custom Check Icon

CheckBox check icon can be customized as per the requirement by adding CSS rules.

In the following example, the check icon can be customized by changing check icon content, background and border color in focus and hovered states by adding `e-checkicon` class.

CSHTML

```
@Html.EJS().CheckBox("cbox2").Label("Buy
Groceries").Checked(true).CssClass("e-checkicon").Render()
@Html.EJS().CheckBox("cbox3").Label("Pay Rent").Checked(true).CssClass("e-
checkicon").Render()
@Html.EJS().CheckBox("cbox4").Label("Make Dinner").CssClass("e-
checkicon").Render()
@Html.EJS().CheckBox("cbox1").Label("Finish To-do List
Article").CssClass("e-checkicon").Render()
<style>
    @@font-face {
        font-family: 'btn-icon';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1tSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYagVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAARAAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQON6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaAABAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAACAABAAAAAQAAJ1LUzF8
PPPUACwQAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaawAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAAlIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAAEAAAAAA4sD9AACAAATARF0AxcCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwWMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtOODQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEVBt8LLwg3NT8INS8FNT8NNS8JBjYU/BDUvCyMPAQytrQH5AgoEAQEBArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAgIEBAoBAQEECgCHBgUFBAMDAQEBAQQFBwkFBQUGef6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEEDBgokDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCAcGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBaIFawIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBABAgADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmZ8KNScxBxEfDjsBHQEfdTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQL7Y0EAWQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAChBwYGBgUFBQDAGIBAQEBAgI
DBAQFBQYGBgCHBwgMAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAWI
CUAGHBwcGBgYFBQQEAWMBAgIBAwMEBAUFBQcGBwCHCAImCAChBwYGBgUFBQDAGIBAdUJCQgICAg
GBwYFBAQDAGEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAAABAAAGAAQABAAAAA
CAACACQABAAAAAADAAGAEABAAAAAEEAAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAxWADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
```

```

A1wADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAAYgB0AG4ALQBpAGM
ABwBuAGIAdABuAC0AaQBJAG8AbgBWAGUAcgBzAGkABwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYABwBuAHQAIABnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuaGMAZgB1AHM
AaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkABwB3AHcAdwAuAHMAeQBuaGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
GAQcBCAEJAAPtZWRpYS1wbGF5C21lZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTaxBGNvcHkQLWRvd25sb2FkLTAYLXdmLQAA) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-icons {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
li {
    list-style: none;
}
.e-checkbox-wrapper {
    margin-top: 18px;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check::before {
    content: '\e703';
}
.e-checkicon.e-checkbox-wrapper .e-check {
    font-size: 8px;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check {
    background-color: white;
    border-color: grey;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper:hover .e-frame.e-check {
    background-color: white;
    border-color: grey;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-checkbox:focus + .e-frame.e-check {
    background-color: white;
    border-color: grey;
    box-shadow: none;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-ripple-element {
    background: grey;
}
</style>

```

CUSTOMICON.CS

```
public ActionResult CustomIcon()  
{  
    return View();  
}
```

Note: [View Sample in GitHub.](#)

Name and Value in form submit

The name attribute of the CheckBox is used to group Checkboxes. When the Checkboxes are grouped in form, the checked items value attribute will post to the server on form submit that can be retrieved through the name. The disabled and unchecked CheckBox value will not be sent to the server on form submit.

In the following code snippet, Cricket and Hockey are in the checked state, Tennis is in disabled state and Basketball is in unchecked state. Now, the value that is in checked state only be sent on form submit.

CSHTML

```
@Html.EJS().CheckBox("cbox2").Name("Sport").Value("Cricket").Label("Cricket")  
.Checked(true).Render()  
@Html.EJS().CheckBox("cbox3").Name("Sport").Value("Hockey").Label("Hockey").  
Checked(true).Render()  
@Html.EJS().CheckBox("cbox4").Name("Sport").Value("Tennis").Label("Tennis").  
Render()  
@Html.EJS().CheckBox("cbox1").Name("Sport").Value("Basketball").Label("Basketball").Render()  
@Html.EJS().Button("primarybtn").Content("Submit").IsPrimary(true).Render()  
<style>  
.e-checkbox-wrapper {  
    margin-top: 18px;  
}  
button {  
    margin: 20px 0 0 5px;  
}  
li {  
    list-style: none;  
}  
</style>
```

FORM.CS

```
public ActionResult Form()  
{  
    return View();  
}
```

Note: [View Sample in GitHub.](#)

Right To Left in Check Box Control

CheckBox component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

CSHTML

```
@Html.EJS().CheckBox("cbox2").Label("Default").EnableRtl(true).Render()
<style>
.e-checkbox-wrapper {
    margin-top: 18px;
}
li {
    list-style: none;
}
</style>
```

RTL.CS

```
public ActionResult Rtl()
{
    return View();
}
```

Default ☒

Note: [View Sample in GitHub.](#)

Migration from Essential JS 1

This article describes the API migration process of Checkbox component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Checkbox label | **Property:** *text*

 @Html.EJ().CheckBox("checkbox").Text("Checkbox") | **Property:** *label*

 @Html.EJS().CheckBox("checkbox").Label("Checkbox").Render() |

| Checked state | **Property:** *enableTriState and checkState*

 @Html.EJ().CheckBox("checkbox").Text("Checked state").EnableTriState(true).CheckState(CheckState.Check) | **Property:** *checked*

 @Html.EJS().CheckBox("checkbox").Checked(true).Label("Checked state").Render() |

| Indeterminate state | **Property:** *enableTriState and checkState*

 @Html.EJ().CheckBox("checkbox").Text("Indeterminate state").EnableTriState(true).CheckState(CheckState.Indeterminate) | **Property:** *indeterminate*

 @Html.EJS().CheckBox("checkbox").Indeterminate(true).Label("Indeterminate state").Render() |

| Adding custom css class | **Property:** *cssClass*

 @Html.EJ().CheckBox("checkbox").Text("Checkbox").CssClass("custom-class") | **Property:** *cssClass*

 @Html.EJS().CheckBox("checkbox").CssClass("custom-class").Label("Checkbox").Render() |

| Label position | Not applicable | **Property:** *labelPosition*

@Html.EJS().CheckBox("checkbox").Label("Checkbox").LabelPosition(Syncfusion.EJ2.Buttons.LabelPosition.Before).Render() |

| Disabled state | **Property:** *enabled*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").Enabled(false) | **Property:** *disabled*

@Html.EJS().CheckBox("checkbox").Disabled(true).Label("Checkbox").Render() |

| State persistence | **Property:** *enablePersistence*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").EnablePersistence(true) | **Property:** *enablePersistence*

@Html.EJS().CheckBox("checkbox").EnablePersistence(true).Label("Checkbox").Render() |

| RTL | **Property:** *enableRTL*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").EnableRTL(true) | **Property:** *enableRtl*

@Html.EJS().CheckBox("checkbox").EnableRtl(true).Label("Checkbox").Render() |

| HTML Attributes | **Property:** *htmlAttributes*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").HtmlAttributes("") | Not applicable |

| Id property | **Property:** *id*

@Html.EJ().CheckBox("checkbox").Id("sync") | Not applicable |

| Prefix value of Id | **Property:** *idPrefix*

@Html.EJ().CheckBox("checkbox").IdPrefix("JS") | Not applicable |

| Name attribute | **Property:** *name*

@Html.EJ().CheckBox("checkbox").Name("Conformation") | **Property:** *name*

@Html.EJS().CheckBox("checkbox").Name("Conformation").Render() |

| Value attribute | **Property:** *value*

@Html.EJ().CheckBox("checkbox").Name("Conformation").Value("received") | **Property:** *value*

@Html.EJS().CheckBox("checkbox").Name("Conformation").Value("received").Render() |

| Show rounded corner | **Property:** *showRoundedCorner*

@Html.EJ().CheckBox("checkbox").ShowRoundedCorner(true).Text("Checkbox") | Not applicable |

| Size | **Property:** *size*

@Html.EJ().CheckBox("checkbox").Text("Small").Size(Size.Small) | **Property:** *cssClass*

@Html.EJS().CheckBox("checkbox").CssClass("e-small").Label("Checkbox").Render() |

| Validation rules | **Property:** *validationRules*

@Html.EJ().CheckBox("checkbox").ValidationRules(ViewBag.rules) | Not applicable |

| Validation message | **Property:** *validationMessage*

@Html.EJ().CheckBox("checkbox").ValidationRules(ViewBag.rules).ValidationMessage(ViewBag.message) | Not applicable |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroy | **Method:** *destroy*

 @Html.EJ().CheckBox("checkbox").Text("Checkbox")

var checkbox = \$("#checkbox").data("ejCheckBox");
 checkbox.destroy(); | **Method:** *destroy*

 @Html.EJS().CheckBox("checkbox").Label("Checkbox").Render()
var checkbox =
document.getElementById('checkbox').ej2_instances[0];
checkbox.destroy(); |

| Disable the Checkbox | **Method:** *disable*

@Html.EJ().CheckBox("checkbox").Text("Checkbox")
 var checkbox =
\$("#checkbox").data("ejCheckBox");
checkbox.disable(); | Not applicable |

| Enable the Checkbox | **Method:** *enable*

@Html.EJ().CheckBox("checkbox").Text("Checkbox")
 var checkbox =
\$("#checkbox").data("ejCheckBox");
checkbox.enable(); | Not applicable |

| Check state of the Checkbox | **Method:** *isChecked*

@Html.EJ().CheckBox("checkbox").Text("Checkbox")
 var checkbox =
\$("#checkbox").data("ejCheckBox");
checkbox.isChecked(); | Not applicable |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| BeforeChange Event | **Events:** *beforeChange*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").BeforeChange("beforeChange")

function beforeChange(args) {
 / code block /
} | Not
applicable |

| Change Event | **Events:** *change*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").Change("change")

function
change(args) {
 / code block /
} | Events: *change*

@Html.EJS().CheckBox("checkbox").Label("Checkbox").Change("change").Render()

function change(args) {
 / code block /
} |

| created Event | **Events:** *create*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").Create("create")

function
create(args) {
 / code block /
} | Events: *created*

@Html.EJS().CheckBox("checkbox").Label("Checkbox").Created("created").Render()

function created() {
 / code block /
} |

| Destroy Event | **Events:** *destroy*

@Html.EJ().CheckBox("checkbox").Text("Checkbox").Destroy("destroy")

function
destroy(args) {
 / code block /
} | Not applicable |

Chips

Getting Started with ASP.NET MVC Chip Control

This section briefly explains about how to include [ASP.NET MVC Chip](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC

controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Chip control

Now, add the Syncfusion ASP.NET MVC Chip control in `~/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().ChipList("chip").Text("Janet Leverling").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Chip control will be rendered in the default web browser.

Janet Leverling

Note: [View Sample in GitHub.](#)

Types in Chips Control

The `ChipList` control has the following types.

- Input Chip
- Choice Chip
- Filter Chip
- Action Chip

Input Chip

Input Chip holds information in compact form. It converts user input into chips.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Chips(chip =>
{
    chip.Text("Andrew").Enabled(true).Add();
    chip.Text("Janet").Enabled(true).Add();
    chip.Text("Laura").Enabled(true).Add();
    chip.Text("Margaret").Enabled(true).Add();
}).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Choice Chip

Choice Chip allows to select a single chip from the set of ChipList/ChipCollection. It can be enabled by setting the **selection** property to **Single**.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Selection(Selection.Single).Chips(chip
=>
    {
        chip.Text("Small").Enabled(true).Add();
        chip.Text("Medium").Enabled(true).Add();
        chip.Text("Large").Enabled(true).Add();
        chip.Text("Extra Large").Enabled(true).Add();
    }).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Filter Chip

Filter Chip allows to select a multiple chip from the set of ChipList/ChipCollection. It can be enabled by setting the **selection** property to **Multiple**.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Selection(Selection.Multiple).Chips(chip
=>
    {
        chip.Text("Chai").Enabled(true).Add();
        chip.Text("Chang").Enabled(true).Add();
        chip.Text("Aniseed Syrup").Enabled(true).Add();
        chip.Text("Ikura").Enabled(true).Add();
    }).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Action Chip

The Action Chip triggers the event like click or delete, which helps doing action based on the event.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Chips(chip =>
{
    chip.Text("Send a text").Enabled(true).Add();
    chip.Text("Set a
remainder").Enabled(true).Add();
    chip.Text("Read my emails").Enabled(true).Add();
    chip.Text("Set alarm").Enabled(true).Add();
}).Click("chipclick").Render()

<script>
function chipclick(e) {
    alert('you have clicked ' + e.text);
}
</script>
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Deletable Chip

Deletable Chip allows to delete a chip from ChipList/ChipCollection. It can be enabled by setting the `enableDelete` property to `true`.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").EnableDelete(true).Chips(chip =>
{
    chip.Text("Send a text").Enabled(true).Add();
    chip.Text("Set a
remainder").Enabled(true).Add();
    chip.Text("Read my emails").Enabled(true).Add();
    chip.Text("Set alarm").Enabled(true).Add();
}).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Chip Customization in Chip Control

This section explains the customization of styles, leading icons, avatar, and trailing icons in Chip control.

Styles

The Chip control has the following predefined styles that can be defined using the `cssClass` property.

Class	Description
-------	-------------

- | ----- | ----- |
- | e-primary | Represents a primary chip. |
- | e-success | Represents a positive chip. |
- | e-info | Represents an informative chip. |
- | e-warning | Represents a chip with caution. |
- | e-danger | Represents a negative chip. |

CSHTML

```
@Html.EJS().ChipList("chip-avatar").EnableDelete(true).Chips(chip =>
{
    chip.Text("Primary").CssClass("e-
primary").Enabled(true).Add();
    chip.Text("Success").CssClass("e-
success").Enabled(true).Add();
    chip.Text("Info").CssClass("e-
info").Enabled(true).Add();
    chip.Text("Warning").CssClass("e-
warning").Enabled(true).Add();
    chip.Text("Danger").CssClass("e-
danger").Enabled(true).Add();
}).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Leading Icon

You can add and customize the leading icon of chip using the `leadingIconCss` property.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Chips(chip =>
{
    chip.Text("Andrew").LeadingIconCss("andrew").Enabled(true).Add();
    chip.Text("Janet").LeadingIconCss("janet").Enabled(true).Add();
    chip.Text("Laura").LeadingIconCss("laura").Enabled(true).Add();
    chip.Text("Margaret").LeadingIconCss("margaret").Enabled(true).Add();
}).Render()

<style>
#chip-avatar .andrew {
    background-image: url('../andrew.png')
}
#chip-avatar .margaret {
    background-image: url('../margaret.png')
```

```
}
#chip-avatar .laura {
    background-image: url('./laura.png')
}
#chip-avatar .janet {
    background-image: url('./janet.png')
}
</style>
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Avatar

You can add and customize the avatar of chip using the `avatarIconCss` property.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Chips(chip =>
{
    chip.Text("Andrew").AvatarIconCss("andrew").Enabled(true).Add();
    chip.Text("Janet").AvatarIconCss("janet").Enabled(true).Add();
    chip.Text("Laura").AvatarIconCss("laura").Enabled(true).Add();
    chip.Text("Margaret").AvatarIconCss("margaret").Enabled(true).Add();
}) .Render()
<style>
#chip-avatar .andrew {
    background-image: url('./andrew.png')
}
#chip-avatar .margaret {
    background-image: url('./margaret.png')
}
#chip-avatar .laura {
    background-image: url('./laura.png')
}
#chip-avatar .janet {
    background-image: url('./janet.png')
}
</style>
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```


Avatar Content

You can add and customize the avatar content of chip using the `avatarText` property.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Chips(chip =>
{
    chip.Text("Andrew").AvatarText("A").Enabled(true).Add();
    chip.Text("Janet").AvatarText("J").Enabled(true).Add();
    chip.Text("Laura").AvatarText("L").Enabled(true).Add();
    chip.Text("Margaret").AvatarText("M").Enabled(true).Add();
}).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Trailing Icon

You can add and customize the trailing icon of chip using the `trailingIconCss` property.

CSHTML

```
@Html.EJS().ChipList("chip-avatar").Chips(chip =>
{
    chip.Text("Andrew").TrailingIconCss("e-dlt-
btn").Enabled(true).Add();
    chip.Text("Janet").TrailingIconCss("e-dlt-
btn").Enabled(true).Add();
    chip.Text("Laura").TrailingIconCss("e-dlt-
btn").Enabled(true).Add();
    chip.Text("Margaret").TrailingIconCss("e-dlt-
btn").Enabled(true).Add();
}).Render()
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Outline Chip

Outline chip has the border with the background transparent. It can be set using the `cssClass` property.

CSHTML

```
<div>
```

```
@Html.EJS().ChipList("chip-avatar").CssClass("e-outline").Chips(chip =>
{
    chip.Text("Chai").Enabled(true).Add();
    chip.Text("Chung").Enabled(true).Add();
    chip.Text("Aniseed Syrup").Enabled(true).Add();
    chip.Text("Ikura").Enabled(true).Add();
}).Render()
@Html.EJS().ChipList("chip-avatar").CssClass("e-
outline").EnableDelete(true).Chips(chip =>
{
    chip.Text("Andrew").Enabled(true).Add();
    chip.Text("Janet").Enabled(true).Add();
    chip.Text("Laura").Enabled(true).Add();
    chip.Text("Margaret").Enabled(true).Add();
}).Render()
</div>
```

DEFAULT.CS

```
public IActionResult Index()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the chip text

Use the following CSS to customize the chip text properties.

```
`css
.e-chip .e-chip-text {
font-size: 20px;
color: black;
font-weight: normal;
}
```

Customizing the chip icon

Use the following CSS to customize the chip icon properties.

```
`css
.e-chip .e-icon {
background-image: url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png');
opacity: 0.8;
```

```
}  
,
```

Customizing the chip delete button

Use the following CSS to customize the chip delete button.

```
`css  
.e-chip-list .e-chip .e-chip-delete.e-dlt-btn {  
  color: #e3165b;  
  font-size: 12px;  
}  
,
```

Customizing the chip outline

Use the following CSS to customize the chip outline.

```
`css  
.e-chip-list .e-chip.e-outline {  
  border-color: #e3165b;  
  border-width: 3px;  
}  
,
```

Customizing the chip on selection

Use the following CSS to customize the chip on selection.

```
`css  
  
/ To customize single chip on selection /  
.e-chip-list.e-selection .e-chip.e-active {  
  background-color: #ffca1c;  
  color: #e3165b;  
}  
  
/ To customize multiple chip on selection /  
.e-chip-list .e-chip.e-active {  
  background-color: #e3165b;  
  color: white;  
}  
,
```

Customizing the chip avatar text

Use the following CSS to customize the chip avatar text properties.

```
`css
.e-chip-list .e-chip .e-chip-avatar {
background-color: #d51a1a;
color: #fafafa;
}
`
```

Accessibility in ASP.NET MVC Chips component

The Chips component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Chips component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Chips component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Chips component:

| Attributes | Purpose |

| --- | --- |

| **role=checkbox** | Indicates the ChipList component wrapper element as **checkbox**. |

| **role=option** | Used to convey a significant and contextual message to the user(ChipList). |

| **role=button** | Used to convey a significant and contextual message to the user(Single Chip). |

| **aria-label** | Provides an accessible name for the Chip. |

| **aria-selected** | Indicates the element is selected. |

| **aria-disabled** | Indicates element is perceivable but disabled. |

| **aria-multiselectable** | Indicates multiple items to be selected. |

Keyboard interaction

The Chips component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Chips component.

| Keyboard shortcuts | Actions |

| ----- | ----- |

| **Enter / Space** | Selects the targeted chip from the ChipList/ChipCollection. |

| **Delete / Backspace** | Deletes the targeted chip from the ChipList/ChipCollection. |

Ensuring accessibility

The Chips component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Chips component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Chips component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

CircularGauge

Getting Started with ASP.NET MVC Circular Gauge Control

This section briefly explains about how to include [ASP.NET MVC Circular Gauge](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{ site.ej2version
  }/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Circular Gauge control

Now, add the Syncfusion ASP.NET MVC Circular Gauge control in `~/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().CircularGauge("container").Render();
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Circular Gauge control will be rendered in the default web browser.



Add Gauge Title

You can add a title using [Title](#) attribute to the circular gauge to provide quick information to the user.

CSHTML

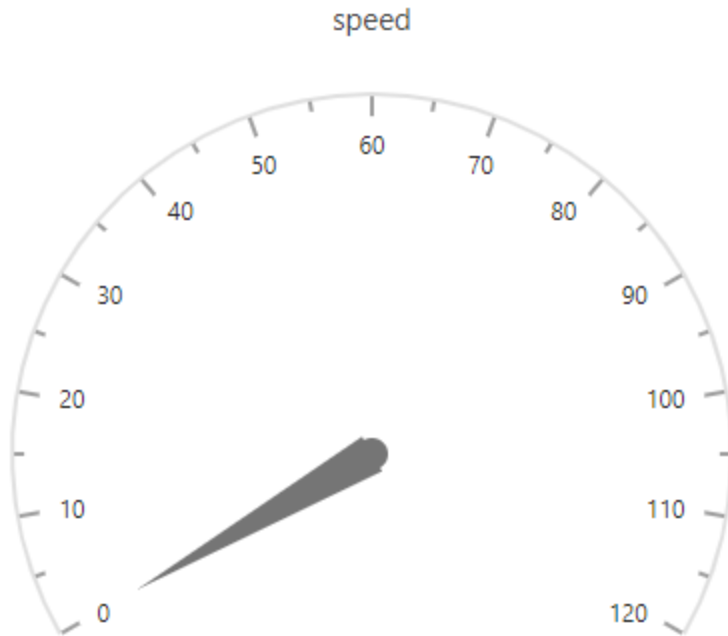
```
@Html.EJS().CircularGauge("container").Title("Speed").Render();
```

Axis

You can set the range to the axis using [Minimum](#) and [Maximum](#) attributes for axis tag. Refer below code snippet to add the axis range to circular gauge.

CSHTML

```
@Html.EJS().CircularGauge("container").Title("Speed").Axes(axes =>  
    axes.EndAngle(120).StartAngle(240).Radius("90%").Minimum(0).Maximum(120).Add  
   ()).Render();
```

Note: [View Sample in GitHub.](#)

Circular Gauge Dimensions

Size for Circular Gauge

<!-- markdownlint-disable MD036 -->

You can also set size for the gauge directly through [width](#) and [height](#) properties.

In Pixel

You can set the size of the gauge in pixel as demonstrated below.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular").Height("350px").Width("650px").Render(  
)
```

GAUGE-DIMENSION.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            // ...  
        }  
    }  
}
```

```

        {
            return View();
        }
    }
}

```

In Percentage

By setting value in percentage, gauge gets its dimension with respect to its container. For example, when the height is '50%', gauge renders to half of the container height.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Height("50%").Width("80%").Render()

```

PERCENTAGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Note: [View Sample in GitHub](#).

Axes in Circular Gauge Control

By default, gauge will be displayed with an axis. Each axis contains its own ranges, pointers and annotation.

<!-- markdownlint-disable MD036 -->

Axis Customization

You can customize the width and color of an axis line by using [LineStyle](#) property. Background for an axis can be customized by using [background](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes => axes.LineStyle(ls
=>ls.Color("red").Width(2))

```

```
.Background("rgba(0, 128, 128, 0.3)").Add()).Render()
```

AXES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Angles and Direction

Circular gauge axis can sweep from 0 to 360 degrees. By default start angle of an axis is 200 degree and end angle is 160 degree and you can customize this option by using [startAngle](#) and [endAngle](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes=>axes.StartAngle(270).EndAngle(90).Add()).Render()
```

ANGLES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

The [direction](#) property enables you to render the gauge axis either in **ClockWise** or in **AntiClockWise** direction.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular").Axes(axes=>axes.Direction(Syncfusion.EJ2.CircularGauge.GaugeDirection.AntiClockWise).Add()).Render()
```

DIRECTION.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            return View();  
        }  
    }  
}
```

Axis Radius

By default, radius of an axis is calculated based on the available size. You can customize this, by using [radius](#) property. It takes value either in **percentage** or in **pixel**.

In Pixel

You can set the radius of the gauge in pixel as demonstrated below,

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular").Axes(axes=>axes.Radius("150").Add()).Render()
```

RADIUS-PIXEL.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{
```

```

public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}

```

In Percentage

By setting value in percentage, gauge gets its dimension with respect to its available size. For example, when the radius is '50%', gauge renders to half of the available size.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes=>axes.Radius("50%").Add()).Render()

```

RADIUS-PERCENTAGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Ticks

You can customize the [height](#), [color](#) and [width](#) of major and minor ticks by using [majorTicks](#) and [minorTicks](#) property. By default, [interval](#) for [majorTicks](#) will be calculated automatically and also you can customize the interval for major and minor ticks using [interval](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes => axes
.MajorTicks(majortick =>
majortick.Interval(10).Color("red").Height(10).Width(3))
.MinorTicks(minortick
=>minortick.Interval(5).Color("green").Height(5).Width(2))
.Add()).Render()

```

TICKS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Tick Position

Both minor and major ticks can be moved by using [offset](#) and [position](#) property. The [offset](#) defines the distance between the axis and ticks. By default, offset value is 0. The [position](#) will place the ticks either inside or outside of the axis. By default, ticks will be placed **inside** the axis.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular2").Axes(axes => axes
    .MajorTicks(majortick =>
        majortick.Interval(10).Color("red").Height(10).Width(3).Position(Position.Inside).Offset(5))
    .MinorTicks(minortick
        =>minortick.Interval(10).Color("green").Height(5).Width(2).Position(Position.Inside).Offset(5))
    .Add()).Render()
```

TICK-POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {

```

```

        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Labels

Labels of an axis can be customized by using [font](#) property in [labelStyle](#) options.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular2").Axes(axes => axes
.LabelStyle(ls => ls.Font(new CircularGaugeFont {Color="red", Size="20px",
FontWeight="Bold" })).Add()).Render()

```

LABELS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Label Position

Labels can be moved by using [offset](#) or [position](#) property. The [offset](#) defines the distance between the labels and ticks. By default, offset value is 0. The [position](#) will place the labels either inside or outside of the axis. By default, labels will be placed inside the axis.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
.LabelStyle(labels
=>labels.Position(Position.Outside).Offset(5)).Add()).Render()

```

LABEL-POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Auto Angle

Labels can be swept along the axis angle by enabling [autoAngle](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
.LabelStyle(labels => labels.AutoAngle(true)).Add()).Render()
```

AUTO-ANGLES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Smart Labels

When an axis makes a complete circle, then the first and last label of the axis will get overlap with each other. In this scenario, you can either hide 1st or last label using [hiddenLabel](#) property. When

`hiddenLabel` value is `First`, then the 1st label will be hidden and when the `hiddenLabel` value is `'Last'`, then the last label will be hidden.

CSHTML

```
using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes =>
axes.Minimum(0).Maximum(12).StartAngle(0).EndAngle(360)
.MajorTicks(majorticks => majorticks.Interval(1)
.Position(Position.Inside).Height(10))
.MinorTicks(minorticks => minorticks.Interval(0.2)
.Position(Position.Inside).Height(5))
.LabelStyle(labels => labels.Position(Position.Inside)
.HiddenLabel(HiddenLabel.First)).Add()).Render()
```

SMART-LABELS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Label Format

Axis labels can be formatted by using [format](#) property in [labelStyle](#) and its supports all globalize format.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
.LabelStyle(labels => labels.Format("p1")).Add()).Render()
```

LABEL-FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```

using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal places
1000	n3	1000.000	The Number is rounded to 3 decimal places
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal places
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal places
1000	c1	\$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal places

Custom Label Format

Axis labels support custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .LabelStyle(labels
=>labels.Format("{value}°C")).Add()).Render()

```

CUSTOM-LABEL-FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Hide Intersecting Axis Labels

When the axis labels overlap with each other, you can hide the intersected labels by setting the `hideIntersectingLabel` property to true in the axis.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").Axes(axes => axes
    .Minimum(0).Maximum(200).StartAngle(270).EndAngle(90).HideIntersectingLabel(
true)
    .MajorTicks(mi => mi.Interval(4)).MinorTicks(mj =>
mj.Interval(2)).Add()).Render()
```

HIDE-INTERSECTING-LABEL.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Minimum and Maximum

The [minimum](#) and [maximum](#) properties enables you to customize the start and end values of an axis.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular").Axes(axes =>  
axes.Minimum(50).Maximum(250).Add()).Render()
```

MINMAX.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            return View();  
        }  
    }  
}
```

Multiple Axes

In addition to the default axis, you can add n number of axis to a gauge. Each axis will have its own ranges, pointers, annotations and customization options.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular2").Axes(axes => {  
    axes  
    .MajorTicks(majortick =>  
majortick.Interval(10).Color("red").Height(10).Width(3))  
    .MinorTicks(minortick =>  
minortick.Interval(5).Color("green").Height(5).Width(2))  
    .Add();  
    axes.MajorTicks(majortick =>  
majortick.Interval(10).Color("blue").Height(10).Width(3))  
    .MinorTicks(minortick =>  
minortick.Interval(5).Color("lime").Height(5).Width(2)).Add();  
}).Render()
```

MULTIPLE-AXES.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.CircularGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Ranges in Circular Gauge Control

You can categories certain interval on gauge axis using [ranges](#) property.

Start and End

Start and end value of a range in an axis can be customized by using [start](#) and [end](#) properties.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>
    axes.Ranges(range =>range.Start(20).End(80).Add())
    .Add()).Render()
```

RANGES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Customization

Color and thickness of the range can be customized by using [color](#), [startWidth](#) and [endWidth](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>
    axes.Ranges(range
=>range.Start(20).End(80).StartWidth(15).EndWidth(15).Color("#ff5985").Add()
)
.Add()).Render()
```

CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

<!-- markdownlint-disable MD036 -->

Radius

You can place the range inside or outside of the axis by using [radius](#) property. The radius of the range can take value either in percentage or in pixels. By default, ranges take 100% of the axis radius.

In Pixel

You can set the radius of the range in pixel.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>
    axes.Ranges(range =>range.Start(20).End(80).Radius("100").Add()
)
.Add()).Render()
```

RADIUS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

<!-- markdownlint-disable MD036 -->

In Percentage

By setting value in percentage, range gets its dimension with respect to its axis radius. For example, when the radius is '50%', range renders to half of the axis radius.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>
    axes.Ranges(range =>range.Start(20).End(80).Radius("60%").Add())
    .Add()).Render()
```

RADIUS-PERCENTAGE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Dragging Ranges

The ranges can be dragged over the axis line by clicking and dragging the same. To enable or disable the range drag, use the [EnableRangeDrag](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Height("250px").Width("250px").EnableRangeDrag(true).Axes(axes => axes
    .Pointers(pointer => pointer.Value(50)).Ranges(range
=>range.Start(0).End(100).StartWidth(8).EndWidth(8).Radius("108%").Color("#30B32D").Add())
    .Add()).Render()
```

DRAGGING-RANGE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Multiple Ranges

You can add multiple ranges to an axis with the above customization.

Note: You can set the range color to axis ticks and labels by enabling `useRangeColor` property in [majorTicks](#), [minorTicks](#) and [labelStyle](#) object.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>
    axes.Minimum(0).Maximum(100).MajorTicks(mt =>mt.UseRangeColor(true)).
    MinorTicks(minor=> minor.UseRangeColor(true))
    .LabelStyle(labels =>labels.UseRangeColor(true))
    .Ranges(range => {
        range.Start(0).End(25).Radius("108%").Add();
        range.Start(25).End(50).Radius("70%").Add();
        range.Start(50).End(75).Radius("70%").Add();
        range.Start(75).End(100).Radius("108%").Add();
    })
    .Add()).Render()
```

MULTIPLE-RANGES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```



```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Rounded corner radius

You can customize the corner radius using the `roundedCornerRadius` property in `ranges`.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>
    axes.Ranges(range =>
        range.Start(20).End(80).Radius("60%").RoundedCornerRadius(6).Add())
    .Add()).Render()
```

RANGES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Gradient Color

Gradient support allows to add multiple colors in the ranges and pointers of the circular gauge. The following gradient types are supported in the circular gauge.

- Linear Gradient

- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient will be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

CSHTML

```
@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").Load("onGaugeLoad").Axes(axes => axes
    .Radius("80%").StartAngle(200).EndAngle(160).Minimum(0).Maximum(100).LabelSt
yle(ls => ls.Position(Position.Inside))
    .MajorTicks(major => major.Height(0)).MinorTicks(minor =>
minor.Height(0))
    .Pointers(pt =>
pt.Value(60).RoundedCornerRadius(20).PointerWidth(0).Cap(cap =>
cap.Radius(0)).Add())
    .LineStyle(lin => lin.Width(0))
    .Add()).Render()
<script>
    var rangeLinearGradient = {
        startValue: '0%',
        endValue: '100%',
        colorStop: [
            { color: '#9E40DC', offset: '0%', opacity: 0.9 },
            { color: '#E63B86', offset: '70%', opacity: 0.9 }
        ]
    }
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].minorTicks.height = 0;
        sender.gauge.axes[0].majorTicks.height = 0;
        sender.gauge.axes[0].labelStyle.fontSize = "0px";
        sender.gauge.axes[0].ranges = [{
            start: 0,
            end: 100,
            radius: '90%',
            startWidth: 30,
            endWidth: 30,
            color: '#E0E0E0',
            roundedCornerRadius: 20,
            linearGradient: rangeLinearGradient
        }];
    }
</script>
```

LINEARGRADIENT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
```

```

using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

CSHTML

```

@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").Load("onGaugeLoad").Axes(axes => axes
    .Radius("80%").StartAngle(200).EndAngle(160).Minimum(0).Maximum(100).LabelSt
yle(ls => ls.Position(Position.Inside))
    .MajorTicks(major => major.Height(0)).MinorTicks(minor =>
minor.Height(0))
    .Pointers(pt =>
pt.Value(60).RoundedCornerRadius(20).PointerWidth(0).Cap(cap =>
cap.Radius(0)).Add())
    .LineStyle(lin => lin.Width(0))
    .Add()).Render()
<script>
    var rangeRadialGradient = {
        radius: '50%', innerPosition: { x: '50%', y: '50%' },
        outerPosition: { x: '50%', y: '50%' },
        colorStop: [
            { color: '#9E40DC', offset: '90%', opacity: 0.9 },
            { color: '#E63B86', offset: '160%', opacity: 0.9 } ]
    }
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].minorTicks.height = 0;
        sender.gauge.axes[0].majorTicks.height = 0;
        sender.gauge.axes[0].labelStyle.font.size = "0px";
        sender.gauge.axes[0].ranges = [{
            start: 0,
            end: 100,
            radius: '90%',
            startWidth: 30,
            endWidth: 30,
            color: '#E0E0E0',
            roundedCornerRadius: 20,
            radialGradient: rangeRadialGradient
        }];
    }

```

```
    }];
}
</script>
```

RADIALGRADIENT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

See also

- [Tooltip for Ranges](#)

Pointers in Circular Gauge Control

Pointers are used to indicate values on the axis. Value of the pointer can be modified using the [value](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
.Pointers(pointer => pointer.Value(90).Add()).Add()).Render()
```

POINTERS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
```

```

{
    public IActionResult Index()
    {
        return View();
    }
}

```

Gauge supports 3 types of pointers such as **Needle**, **RangeBar** and **Marker**. You can choose any one of the pointer by using [type](#) property.

Needle Pointers

A needle pointer contains three parts, a needle, a cap / knob and a tail. The length of the needle can be customized by using [radius](#) property. The length of the tail can be customized by using [length](#) property. The radius of the cap can be customized by using [radius](#) in cap object. The needle and tail length takes value either in **percentage** or **pixel**.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer => pointer.Value(90)
        .Radius("50%")
        .Cap(cab => cab.Radius(10)).NeedleTail(needle =>
needle.Length("25%"))
        .Add())
    .Add()).Render()

```

NEEDLE-POINTERS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

<!-- markdownlint-disable MD036 -->

Customization

Needle color and width can be customized by using [color](#) and [pointerWidth](#) property. Cap and tails can be customized by using [cap](#) and [needleTail](#) object.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer => pointer.Value(90)
        .Radius("50%").Color("#007DD1").PointerWidth(10)
        .Border(border =>
border.Color("#007DD1").Width(5))
        .Cap(cab => cab.Radius(10).Color("white"))
        .NeedleTail(needle =>
needle.Length("25%").Color("#007DD1"))
        .Add())
    .Add()).Render()
```

CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

The appearance of the needle pointer can be customized by using [NeedleStartWidth](#) and [NeedleEndWidth](#).

CSHTML

```
@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes =>
axes.StartAngle(270).EndAngle(90).Minimum(0).Maximum(100)
    .Pointers(pointer => pointer.Value(70)
        .Radius("80%").Color("green").PointerWidth(2).NeedleStartWidth(4).NeedleEndW
idth(4)
        .Animation(animation =>
animation.Enable(true).Duration(1000))
        .Cap(cab => cab.Radius(8).Color("Green"))
        .NeedleTail(needle => needle.Length("0%"))
```

```

        .Add()).LineStyle(l => l.Width(3).Color("#1E7145")).MajorTicks(mjt
=> mjt.Width(1).Height(0).Interval(100))
        .MinorTicks(mit => mit.Width(0).Height(0)).Annotations(annotation =>
annotation.Angle(180).ZIndex("1").Radius("20%")
        .Content("#template").Add())
        .LabelStyle(ls => ls.Position(Position.Outside).Font(new
CircularGaugeFont { Color = "#1E7145", Size = "0px" }))
        .Add()).Render()
<script id='template' type='text/x-template">
    <div id='templateWrap'>
        <div class='des'>
            <div style="color:#757575; font-family:Roboto; font-
size:14px;padding-top: 26px">Customized Needle</div>
        </div>
    </div>
</script>

```

NEEDLE-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

RangeBar Pointer

RangeBar pointer is like ranges in an axis, that can be placed on gauge to mark the pointer value. RangeBar starts from the beginning of the gauge and ends at the pointer value.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer =>
pointer.Value(50).Type(PointerType.RangeBar).Radius("50%").Add())
    .Add()).Render()

```

RANGEBAR-POINTER.CS

```

using System;
using System.Collections.Generic;

```

```

using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Customization

RangeBar can be customized in terms of color, border and thickness by using [color](#), [border](#) and [pointerWidth](#) property.

CSHTML

```

@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer =>
        pointer.Value(50).Type(PointerType.RangeBar).Radius("50%")
        .Color("#007DD1").PointerWidth(15)
        .Border(border =>border.Color("grey").Width(2)).Add())
    .Add()).Render()

```

POINTER-CUSTOMIZATION.CS

```

using Microsoft.AspNetCore.Mvc;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Rounded corner for range bar pointer

The start and end pointers of range bar in the circular gauge are rounded to form arc gauges.

CSHTML

```

@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes => axes

```



```
.Pointers(pointer => pointer.Value(50).Type(PointerType.RangeBar)
.Radius("50%").RoundedCornerRadius(6).Add())
.Add()).Render()
```

RANGEBAR-POINTER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Marker Pointer

Different type of marker shape can be used to mark the pointer value in axis. You can change the marker shape using [markerShape](#) property in pointer. Gauge supports the below marker shape.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond

The image can be used instead of rendering marker shape to denote the pointer value. It can be achieved by setting [markerShape](#) to Image and assigning image path to [imageUrl](#) in pointer.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer => pointer.Value(50).Type(PointerType.Marker)

    .Radius("100%").MarkerShape(GaugeShape.InvertedTriangle).MarkerWidth(15).MarkerHeight(15).Add())
.Add()).Render()
```

MARKER-POINTER.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Customization

The marker can be customized in terms of color, border, width and height by using [color](#), [border](#), [markerWidth](#) and [markerHeight](#) property in [pointer](#).

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer =>
        pointer.Value(50).Type(PointerType.Marker).Color("white")
            .Border(border => border.Color("#007DD1").Width(2))

        .Radius("100%").MarkerShape(GaugeShape.InvertedTriangle).MarkerWidth(15).MarkerHeight(15).Add())
    .Add()).Render()

```

MARKER-CUSTOMIZATION.CS

```

using Microsoft.AspNetCore.Mvc;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Dragging Pointer

The pointers can be dragged over the axis line by clicking and dragging the same. To enable or disable the pointer drag, use the [EnablePointerDrag](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Height("250px").Width("250px").EnablePointerDrag(true).Axes(axes => axes
    .Pointers(pointer => pointer.Value(50)
        .Add())
    .Add()).Render()
```

DRAGGING-POINTER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Multiple Pointers

In addition to the default pointer, you can add n number of pointer to an axis by using **pointers** property.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer => {
        pointer.Value(90).Type(PointerType.Marker)

        .Radius("100%").MarkerShape(GaugeShape.Triangle).MarkerWidth(15).MarkerHeight(15).Add();

        pointer.Value(90).Type(PointerType.RangeBar).Radius("60%").PointerWidth(10).Add();
        pointer.Value(90).NeedleTail(nt => nt.Length("22%")).PointerWidth(20)
            .Cap(cap => cap.Radius(15).Border(border =>
                border.Width(5))).Radius("60%").Add();
    })
    .Add()).Render()
```

MULTIPLE-POINTER.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Animation

Pointer will get animate on loading the gauge, this can be handled by using [animation](#) property in pointer. The [enable](#) property in animation allows to enable or disable the animation. The [duration](#) property specifies the duration of the animation in milliseconds.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").Axes(axes => axes
    .Pointers(pointer => {
        pointer.Value(90).Type(PointerType.Marker).Animation(animation =>
            animation.Enable(true).Duration(500))

        .Radius("100%").MarkerShape(GaugeShape.Triangle).MarkerWidth(15).MarkerHeight(15).Add();

        pointer.Value(90).Type(PointerType.RangeBar).Radius("60%").PointerWidth(10).Add();
        pointer.Value(90).NeedleTail(nt => nt.Length("22%")).PointerWidth(20)
            .Cap(cap => cap.Radius(15).Border(border =>
                border.Width(5))).Radius("60%").Add();
    })
    .Add()).Render()

```

ANIMATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {

```

```

{
    public IActionResult Index()
    {
        return View();
    }
}

```

Gradient Color

Gradient support allows to add multiple colors in the range and pointer of the circular gauge. The following gradient types are supported in the circular gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient will be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

The linear gradient can be applied to all pointer types like marker, range bar and needle.

CSHTML

```

@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").Load("onGaugeLoad").Axes(axes =>
    axes.Radius("90%").StartAngle(270).EndAngle(90)
        .LineStyle(new CircularGaugeLine{ Width = 3, Color =
"#E63B86" }).LabelStyle(new CircularGaugeLabel
    {
        Font = new CircularGaugeFont { Size = "0px" }
    }).MajorTicks(new CircularGaugeTick
    {
        Height = 0,
        Color = "transparent"
    })
    .MinorTicks(new CircularGaugeTick
    {
        Height = 0,
        Color = "transparent"
    })
    .Minimum(0).Maximum(100).Add()).Render()
<script>
    var pointerLinearGradient = {
        startValue: '0%',
        endValue: '100%',
        colorStop: [
            { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
            { color: '#E63B86', offset: '70%', opacity: 0.9 } ]
    }
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].pointers = [{
            radius: '80%',

```

```
        value: 80,
        animation: { enable: true, duration: 1000 },
        pointerWidth: 10,
        linearGradient: pointerLinearGradient,
        cap: {
            radius: 8,
            color: 'white',
            border: {
                color: '#E63B86',
                width: 1
            }
        },
        needleTail: {
            length: '20%',
            linearGradient: pointerLinearGradient,
        }
    }, {
        radius: '60%', value: 40,
        animation: { duration: 1000 },
        pointerWidth: 10,
        linearGradient: pointerLinearGradient,
        cap: {
            radius: 8, color: 'white',
            border: { color: '#E63B86', width: 1 }
        },
        needleTail: {
            length: '20%',
            linearGradient: pointerLinearGradient
        }
    }
    ]];
}
```

</script>

LINEARGRADIENT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

The radial gradient can be applied to all pointer types like marker, range bar and needle.

CSHTML

```
@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").Load("onGaugeLoad").Axes(axes =>
    axes.Radius("90%").StartAngle(270).EndAngle(90)
        .LineStyle(new CircularGaugeLine{ Width = 3, Color =
"#E63B86" }).LabelStyle(new CircularGaugeLabel
    {
        Font = new CircularGaugeFont { Size = "0px" }
    }).MajorTicks(new CircularGaugeTick
    {
        Height = 0,
        Color = "transparent"
    })
    .MinorTicks(new CircularGaugeTick
    {
        Height = 0,
        Color = "transparent"
    })
    .Minimum(0).Maximum(100).Add()).Render()

<script>
    var pointerRadialGradient = {
        radius: '50%',
        innerPosition: { x: '50%', y: '50%' },
        outerPosition: { x: '50%', y: '50%' },
        colorStop: [
            { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
            { color: '#E63B86', offset: '60%', opacity: 0.9 } ]
    }
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].pointers = [{
            radius: '80%',
            value: 80,
            animation: { enable: true, duration: 1000 },
            pointerWidth: 10,
            radialGradient: pointerRadialGradient,
            cap: {
                radius: 8,
                color: 'white',
                border: {
                    color: '#E63B86',
                    width: 1
                }
            },
            needleTail: {
                length: '20%',
                radialGradient: pointerRadialGradient
            }
        }], {
```

```

        radius: '60%', value: 40,
        animation: { duration: 1000 },
        pointerWidth: 10,
        radialGradient: pointerRadialGradient,
        cap: {
            radius: 8, color: 'white',
            border: { color: '#E63B86', width: 1 }
        },
        needleTail: {
            length: '20%',
            radialGradient: pointerRadialGradient
        }
    }];
}
</script>

```

RADIALGRADIENT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

Annotations in Circular Gauge Control

Annotations are used to mark a specific area of interest in the gauge with texts, shapes or images.

Content

You can place any custom element on the axis area by assigning the id of the element to [content](#) property of [annotation](#) object.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes =>
axes.Annotations(annotation => annotation
    .Content("#template").ZIndex("1").Add()).Pointers(pointer =>
pointer.Value(50).Add()).Add()).Render()
<script id='template' type="text/x-template">
    <div id='templateWrap'>

```



```

        <div class='des'>
            <span>Pointer Value : 50</span>
        </div>
    </div>
</script>

```

ANNOTATIONS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Position

Annotation can be placed around the axis by using [radius](#) and [angle](#) property. For example, if the angle is 90 degree and the radius is 110%, then the annotation, will be placed at the right side of the axis.

Radius of the annotation takes value either in pixel or percentage. By setting value in percentage, annotation gets its position with respect to its axis radius.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axes =>
axes.Annotations(annotation => annotation
    .Content("#template").Angle(90).Radius("150%").ZIndex("1").Add())
    .Pointers(pointer => pointer.Value(50).Add()).Add()).Render()
<script id='template' type='text/x-template'>
    <div id='templateWrap'>
        <div class='des'>
            <span>Pointer Value : 50</span>
        </div>
    </div>
</script>

```

POSITION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

```

```

using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Sub Gauge

As the annotation allows to place any custom element, it can initialize a gauge to the element and can be used to place that in another gauge.

CSHTML

```

@using Syncfusion.EJ2.CircularGauge;
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Axes(axis =>
axis.Minimum(0).Maximum(12).LabelStyle(new CircularGaugeLabel
{
    HiddenLabel = HiddenLabel.First
}).LineStyle(new CircularGaugeLine
{
    Width = 0
}).StartAngle(0).EndAngle(360).Annotations(ViewBag.annotations).Pointers(ViewBag.pointers).Ranges(ViewBag.ranges).Add()).Render()
@Html.EJS().CircularGauge("subGauge").Axes(subaxis =>
subaxis.Minimum(0).Maximum(12).LabelStyle(new CircularGaugeLabel
{
    HiddenLabel = HiddenLabel.First,
    Offset = -5
}).MajorTicks(new CircularGaugeTick
{
    Interval = 3
}).LineStyle(new CircularGaugeLine
{
    Width = 0
}).StartAngle(0).EndAngle(360).Pointers(ViewBag.subgauge_pointers).Ranges(ViewBag.subgauge_ranges).Add()).Render()

```

SUBGAUGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;

```

```

using Newtonsoft.Json;
using Syncfusion.EJ2.CircularGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<CircularGaugeRange> ranges = new
List<CircularGaugeRange>();
            CircularGaugeRange range1 = new CircularGaugeRange();
            range1.Start = 0;
            range1.End = 3;
            range1.Color = "rgba(29,29,29,0.7)";
            ranges.Add(range1);
            CircularGaugeRange range2 = new CircularGaugeRange();
            range2.Start = 3;
            range2.End = 12;
            range2.Color = "rgba(168,145,102,0.1)";
            ranges.Add(range2);
            ViewBag.ranges = ranges;
            List<CircularGaugeAnnotation> annotations = new
List<CircularGaugeAnnotation>();
            CircularGaugeAnnotation annotation1 = new
CircularGaugeAnnotation();
            annotation1.Angle = 270;
            annotation1.Radius = "40%";
            annotation1.Content = "<div id=subGauge2 style=width: 90px;
height: 90px;></div>";
            annotations.Add(annotation1);
            CircularGaugeAnnotation annotation2 = new
CircularGaugeAnnotation();
            annotation2.Angle = 90;
            annotation2.Radius = "40%";
            annotation2.Content = "<div id=time><span>6:30 PM</span></div>";
            annotations.Add(annotation2);
            ViewBag.annotations = annotations;
            List<CircularGaugePointer> pointers = new
List<CircularGaugePointer>();
            CircularGaugePointer pointer1 = new CircularGaugePointer();
            pointer1.PointerWidth = 5;
            pointer1.Radius = "40%";
            pointer1.Value = 6.5;
            pointer1.Border = new CircularGaugeBorder
            {
                Width = 1,
                Color = "rgb(29,29,29)"
            };
            pointer1.Cap = new CircularGaugeCap
            {
                Color = "rgb(29,29,29)",
                Radius = 0,
                Border = new CircularGaugeBorder
                {
                    Width = 0.2,
                    Color = "red"
                }
            }
        }
    }
}

```

```
};
pointer1.NeedleTail = new CircularGaugeNeedleTail
{
    Length = "0%"
};
pointer1.Animation = new CircularGaugeAnimation { Enable = false
};

pointers.Add(pointer1);
CircularGaugePointer pointer2 = new CircularGaugePointer();
pointer2.PointerWidth = 5;
pointer2.Radius = "60%";
pointer2.Color = "rgb(29,29,29)";
pointer2.Value = 6;
pointer2.Border = new CircularGaugeBorder
{
    Width = 1,
    Color = "rgb(29,29,29)"
};
pointer2.Cap = new CircularGaugeCap
{
    Color = "rgb(29,29,29)",
    Radius = 0,
    Border = new CircularGaugeBorder
    {
        Width = 0.2,
        Color = "red"
    }
};
pointer2.NeedleTail = new CircularGaugeNeedleTail
{
    Length = "0%"
};
pointer2.Animation = new CircularGaugeAnimation{ Enable = false
};

pointers.Add(pointer2);
CircularGaugePointer pointer3 = new CircularGaugePointer();
pointer3.PointerWidth = 4;
pointer3.Radius = "70%";
pointer3.Color = "rgba(168,145,102,1)";
pointer3.Value = 9.8;
pointer3.Cap = new CircularGaugeCap
{
    Color = "rgba(168,145,102,1)",
    Radius = 4,
    Border = new CircularGaugeBorder
    {
        Width = 0.2,
        Color = "rgba(168,145,102,1)"
    }
};
pointer3.NeedleTail = new CircularGaugeNeedleTail
{
    Color = "rgba(168,145,102,1)",
    Length = "20%"
};
pointer3.Animation = new CircularGaugeAnimation { Enable =
false, Duration = 0 };
```

```
        pointers.Add(pointer3);
        ViewBag.pointers = pointers;
        List<CircularGaugeRange> subgaugeranges = new
List<CircularGaugeRange>();
        CircularGaugeRange ranges4 = new CircularGaugeRange();
        ranges4.Start = 0;
        ranges4.End = 3;
        ranges4.StartWidth = "5";
        ranges4.EndWidth = "5";
        ranges4.Color = "rgba(29,29,29,0.7)";
        subgaugeranges.Add(ranges4);
        CircularGaugeRange ranges5 = new CircularGaugeRange();
        ranges5.Start = 3;
        ranges5.End = 12;
        ranges5.StartWidth = "5";
        ranges5.EndWidth = "5";
        ranges5.Color = "rgba(168,145,102,0.1)";
        subgaugeranges.Add(ranges5);
        ViewBag.subgauge_ranges = subgaugeranges;
        List<CircularGaugePointer> subpointers = new
List<CircularGaugePointer>();
        CircularGaugePointer pointer4 = new CircularGaugePointer();
        pointer4.PointerWidth = 2;
        pointer4.Radius = "40%";
        pointer4.Animation = new CircularGaugeAnimation { Enable = false
};

        pointer4.Color = "rgb(29,29,29)";
        pointer4.Border = new CircularGaugeBorder
        {
            Width = 1,
            Color = "rgb(29,29,29)"
        };
        pointer4.Cap = new CircularGaugeCap
        {
            Color = "rgb(29,29,29)",
            Radius = 2,
            Border = new CircularGaugeBorder
            {
                Width = 0.2,
                Color = "red"
            }
        };
        pointer4.NeedleTail = new CircularGaugeNeedleTail
        {
            Length = "0%"
        };
        subpointers.Add(pointer4);
        ViewBag.subgauge_pointers = subpointers;
        return View();
    }
}
```

Note: [View Sample in GitHub.](#)

See also

- [Tooltip for Annotations](#)

Animation in Circular Gauge component

All of the elements in the Circular Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the [AnimationDuration](#) property. The animation for the Circular Gauge is enabled when the [AnimationDuration](#) property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the component. If the [AnimationDuration](#) property is set to **0**, which is the default value, the animation effect is disabled. If the animation is enabled, the component will behave in the following order.

1. The axis line will be animated in the rendering direction (clockwise or anticlockwise).
2. Each tick line and label will then be animated.
3. If available, ranges will be animated.
4. If available, pointers will be animated in the same way as [pointer animation](#).
5. If available, annotations will be animated.

The animation of the Circular Gauge is demonstrated in the following example.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("container").AnimationDuration(2000).Load("gaugeLoad").Background("transparent").Axes(axes => axes
    .Radius("80%").StartAngle(230).EndAngle(130).LabelStyle(labelStyle => labelStyle.Position(Position.Inside).Font(font => font.FontFamily("inherit")).Offset(-1))
    .Annotations(annotation => annotation.Content("<div style='font-size:18px;margin-left: -20px;margin-top: -12px;color:#9DD55A'>60</div>").ZIndex("1").Angle(165).Radius("35%").Add())
    .Pointers(pointer => pointer.Value(60).Radius("60%").PointerWidth(7).Color("#c06c84").Animation(animation => animation.Enable(true).Duration(2000)).Cap(cap => cap.Radius(8).Color("#c06c84").Border(border => border.Width(0))).NeedleTail(needleTail => needleTail.Length("0%")).Add())
    .Ranges(range => range.Start(0).End(30).StartWidth(22).EndWidth(22).Color("#E63B86").Radius("60%").Add())
    .LineStyle(lineStyle => lineStyle.Width(8).Color("#E0E0E0"))
    .MinorTicks(minorTick => minorTick.Offset(5)).MajorTicks(majorTick => majorTick.Offset(5)).Add()).Render()
<script>
    var rangeLinearGradient = {
        startValue: '0%',
        endValue: '100%',
        colorStop: [
            { color: '#9e40dc', offset: '0%', opacity: 1 },
            { color: '#d93c95', offset: '70%', opacity: 1 },
        ],
    };
    window.gaugeLoad = function (args) {
```

```
args.gauge.axes[0].ranges[0].linearGradient = rangeLinearGradient;
}
</script>
```

ANIMATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Legend in Circular Gauge Control

Legend provides valuable information for interpreting what the circular gauge axis range displays, and they can be represented in various colors, shapes, and other identifiers based on the data. It gives a breakdown of what each symbol represents in the axis range of circular gauge.

You can add the legend for circular gauge ranges by setting the visible property of `legendSettings` to true.

<!-- markdownlint-disable MD036 -->

Legend customization

Customization option is also provided for the legend shape, alignment, and position.

Position and alignment

The position of the legend is used to place legend in various positions. You can use the `position` property in `legendSettings`. Based on the position, the legend item will be aligned. The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Custom
- Auto

The legend alignment is used to align the legend items in specific location. You can use the alignment property in `legendSettings` to align the legend items. The following options are available to customize the legend alignment:

- Near
- Center
- Far

The legends can also be positioned to absolute position using the `location.x` and `location.y` properties available in `legendSettings`.

Legend size

The legend size can be modified using the `height` and `width` properties in `legendSettings`.

Legend opacity

To specify the transparency for legend shape, set the `opacity` property in `legendSettings`.

Legend shape

To change the legend item shape, specify the desired `shape` in the shape property of the legend. By default, the shape of the legend is `circle`.

It also supports the following shapes:

- Circle
- Rectangle
- Diamond
- Triangle
- InvertedTriangle
- Image

You can customize a shape using the `shapeWidth` and `shapeHeight` properties.

Legend padding

You can control the spacing between the legend items using the `padding` option of the legend. The default value of padding is 5.

Legend border

You can customize the legend border using the `border` option in the legend. The legend border can be customized using the border `color` and `width` properties.

Font of the legend text

The `font` of the legend item text can be customized using the following properties:

- `fontFamily`
- `fontStyle`
- `fontWeight`
- `opacity`
- `color`
- `size`

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("container").LegendSettings(legend =>
    legend.Visible(true)
    .ShapeWidth(30)
    .ShapeHeight(30)
    .Padding(15)
    .Border(border => border.Color("green").Width(3))
    ).Axes(axes => axes.Minimum(0).Maximum(100).LabelStyle(ls =>
ls.UseRangeColor(true))
    .Ranges(range =>
    {
        range.Start(0).End(25).Radius("108%").Add();
        range.Start(25).End(50).Radius("108%").Add();
        range.Start(50).End(75).Radius("108%").Add();
        range.Start(75).End(100).Radius("108%").Add();
    }).MinorTicks(mi => mi.UseRangeColor(true)).MajorTicks(mj =>
mj.UseRangeColor(true)).Add()).Render()
```

LEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

<!-- markdownlint-disable MD036 -->

Toggle option in legend

The toggle option has been provided for legend. So, if you toggle the legend, the given color will be changed to the corresponding circular gauge range. You can enable the toggle option using `toggleVisibility` in the `legendSettings` property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").LegendSettings(legend =>
    legend.Visible(true)
    .ToggleVisibility(true)
```

```

        ).Axes(axes => axes.Minimum(0).Maximum(100).LabelStyle(ls =>
ls.UseRangeColor(true))
        .Ranges(range =>
        {
            range.Start(0).End(25).Radius("108%").Add();
            range.Start(25).End(50).Radius("108%").Add();
            range.Start(50).End(75).Radius("108%").Add();
            range.Start(75).End(100).Radius("108%").Add();
        }).MinorTicks(mi => mi.UseRangeColor(true)).MajorTicks(mj =>
mj.UseRangeColor(true)).Add()).Render()

```

TOGGLE-LEGEND.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

<!-- markdownlint-disable MD036 -->

Paging support in legend

By default, paging will be enabled if the legend items exceed the legend bounds. You can view each legend item by navigating between the pages using navigation buttons.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").LegendSettings(legend =>
    legend.Visible(true)
    .Height("50")
    ).Axes(axes => axes.Minimum(0).Maximum(100).LabelStyle(ls =>
ls.UseRangeColor(true))
    .Ranges(range =>
    {
        range.Start(0).End(25).Radius("108%").Add();
        range.Start(25).End(50).Radius("108%").Add();
        range.Start(50).End(75).Radius("108%").Add();
        range.Start(75).End(100).Radius("108%").Add();
    }).MinorTicks(mi => mi.UseRangeColor(true)).MajorTicks(mj =>
mj.UseRangeColor(true)).Add()).Render()

```

PAGING-SUPPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

<!-- markdownlint-disable MD036 -->

Legend text customization

You can customize the legend text using **legendText** property in **ranges**.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("container").LegendRender("LegendRender").LegendSettings(legend =>
    legend.Visible(true)
    ).Axes(axes => axes.Minimum(0).Maximum(100).LabelStyle(ls =>
ls.UseRangeColor(true))
    .Ranges(range =>
    {
        range.Start(0).End(25).Radius("108%").LegendText("light
air").Add();
        range.Start(25).End(50).Radius("108%").LegendText("light
air").Add();
        range.Start(50).End(75).Radius("108%").LegendText("light
breeze").Add();
        range.Start(75).End(100).Radius("108%").LegendText("light
breeze").Add();
    }).MinorTicks(mi => mi.UseRangeColor(true)).MajorTicks(mj =>
mj.UseRangeColor(true)).Add()).Render()
<script type="text/javascript">
    function LegendRender(args) {
        args.text = 'Legend Modified Text Value'
    }
</script>
```

LEGEND-TEXT-CUSTOM.CS

```
using System;
using System.Collections.Generic;
```

```
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

<!-- markdownlint-disable MD036 -->

legendRendering event will be triggered before rendering each legend item, using this event you can customize needed legend items using following arguments.

Argument Name	Description
fill	Specifies the legend shape color
text	Specifies the current legend text
shape	Customize the shape of the legends
name	Specifies the name of the event
cancel	Set to true, to cancel the event status

Note: [View Sample in GitHub.](#)

User Interaction in Circular Gauge Control

Tooltip for pointers

Circular gauge will display the pointer details through [tooltip](#), when the mouse is moved over the pointer.

<!-- markdownlint-disable MD036 -->

Enable Tooltip

By default, tooltip is not visible. Enable the tooltip by setting [enable](#) property to true.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>axes
    .Pointers(pointer => pointer.Value(70).Add()).Add()).Tooltip(tooltip
=>tooltip.Enable(true)).Render()
```

TOOLTIP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").Axes(axes =>axes
.Pointers(pointer => pointer.Value(70).Add()).Add()).Tooltip(tooltip
=>tooltip.Enable(true).Template("#template")).Render()
<script id='template' type='text/x-template">
    <div id='templateWrap'>
        <div class='des' style='float: right; padding-left:10px; line-
height:30px;'>
            <span>Pointer value : 70</span>
        </div>
    </div>
</script>

```

TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

```
}  
}
```

Tooltip for ranges

Circular gauge displays the information about the ranges through tooltip when hovering the mouse over the ranges. You can enable this feature by setting the type property of tooltip to 'Range' in the array collection.

Tooltip customization for ranges

To customize the range tooltip, use the `rangeSettings` property in tooltip. The following options are available to customize the range tooltip:

- `fill` - Specifies the range tooltip fill color.
- `textStyle` - Specifies the range tooltip text style.
- `format` - Specifies the range content format.
- `template` - Specifies the custom template for tooltip.
- `enableAnimation` - Animates as it moves from one point to another.
- `border` - Specifies the tooltip border.
- `showMouseAtPosition` - Displays the position of the tooltip on the cursor position.

Tooltip for annotations

Circular gauge displays the information about the annotations through tooltip when hovering the mouse over the annotation. You can enable this feature by setting the type property of tooltip to 'Annotation' in the array collection.

Tooltip customization for annotations

To customize the annotation tooltip, use the `annotationSettings` property in tooltip. The following options are available to customize the annotation tooltip:

- `fill` - Specifies the annotation tooltip fill color.
- `textStyle` - Specifies the annotation tooltip text style.
- `format` - Specifies the annotation content format.
- `template` - Specifies the tooltip content with custom template.
- `enableAnimation` - Animates as it moves from one point to another.
- `border` - Specifies the tooltip border.

The following code example shows the tooltip for the pointers, ranges and annotations.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("container").Tooltip(Tl =>  
Tl.Enable(true).Type(new string[] { "Pointer", "Range", "Annotation"  
}).AnnotationSettings(annotation =>  
annotation.Format("CircularGauge")).RangeSettings(range =>  
range.Fill("red"))).Axes(axes => axes
```

```

.Minimum(0).Maximum(120).Radius("90%").StartAngle(240).EndAngle(120).Annotations(
    annotation => annotation.Content("CircularGauge")
    .ZIndex("1").Angle(180).Add()).LabelStyle(ls => ls.UseRangeColor(true)
    .Font(font => font.Color("#424242").Size("13px").FontFamily("Roboto"))
    .Pointers(pt => pt.Value(70).Radius("60%").Color("#33BCBD").Cap(cp =>
    cp.Radius(10).Border(br => br.Width(5).Color("#33BCBD"))
    .Animation(An => An.Enable(false)).Add())
    .LineStyle(lin => lin.Width(0))
    .Ranges(range =>
    {
        range.Start(0).End(50).Radius("102%").StartWidth(10).EndWidth(10).Color("#3A5DC8").Add();

        range.Start(50).End(120).Radius("102%").StartWidth(10).EndWidth(10).Color("#33BCBD").Add();
    })
    .MajorTicks(mi => mi.Color("white").Offset(-5).Height(12)).MinorTicks(mj =>
    mj.Width(0).Color("transparent")).Add()).Render()

```

RANGE-ANNOTATION-TOOLTIP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Pointer Drag

Pointers can be dragged over the axis value. This can be achieved by clicking and dragging the pointer. To enable or disable the pointer drag, you can use [enablePointerDrag](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").EnablePointerDrag(true).Axes(axes
=>axes
.Pointers(pointer => pointer.Value(70).Add()).Add()).Tooltip(tooltip
=>tooltip.Enable(true).Template("#template")).Render()
<script id='template' type='text/x-template">

```

```

        <div id='templateWrap'>
            <div class='des' style="float: right; padding-left:10px; line-
height:30px;">
                <span>Pointer value : 70</span>
            </div>
        </div>
    </script>

```

POINTER-DRAG.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

Print and Export

Print

To use the print functionality, we should set the [AllowPrint](#) property to **true**. The rendered circular gauge can be printed directly from the browser by calling the method [print](#).

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Print").CssClass("e-flat").Render()
@Html.EJS().CircularGauge("circular").Height("450px").AllowPrint(true).Width
("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var circularObj =
document.getElementById('circular').ej2_instances[0];
        circularObj.print();
    };
</script>

```

PRINT.CS

```

using System;

```



```

using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Export

Image Export

To use the image export functionality, we should set the [AllowImageExport](#) property to **true**. The rendered circular gauge can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The circular gauge can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().CircularGauge("circular").Height("450px").AllowImageExport(true)
.Width("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var circularObj =
document.getElementById('circular').ej2_instances[0];
        circularObj.export('PNG', 'CircularGauge');
    };
</script>

```

EXPORT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;

```

```

using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

We can get the image file as base64 string for the JPEG and PNG formats. The circular gauge can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().CircularGauge("circular").Height("450px").AllowImageExport(true)
.Width("650px").Render()
<div id="data"></div>
<script>
    document.getElementById('togglebtn').onclick = function () {
        var circularObj =
document.getElementById('circular').ej2_instances[0];
        var base64 = circularObj.export('JPEG', 'CircularGauge', null,
false).then((data) => {
            document.getElementById("data").innerHTML = data;
        });
    };
</script>

```

EXPORT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller

```

```

    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

PDF Export

To use the PDF export functionality, we should set the [AllowPdfExport](#) property to **true**. The rendered circular gauge can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and "0" indicates portrait and "1" indicates landscape.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().CircularGauge("circular").Height("450px").AllowPdfExport(true).W
idth("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var circularObj =
document.getElementById('circular').ej2_instances[0];
        circularObj.export('PDF', 'CircularGauge', 0);
    };
</script>

```

EXPORT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Note: The exporting of the circular gauge as base64 string is not supported in the PDF export.

Appearance in Circular Gauge Control

Gauge Title

Circular gauge can be given a title by using [title](#) property, to show the information about the gauge. Title can be customized by using [titleStyle](#) property in gauge.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular2").Title("Speedometer").TitleStyle(ts =>  
ts.Color("#27d5ff")).Render()
```

TITLE.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            return View();  
        }  
    }  
}
```

Gauge Position

<!-- markdownlint-disable MD036 -->

Gauge can be positioned anywhere in the container with the help of [centerX](#) and [centerY](#) property and it accepts values either in percentage or in pixels. The default value of the [centerX](#) and [centerY](#) property is 50%, which means gauge will get rendered to the centre of the container.

In Pixel

You can set the mid point of the gauge in pixel as demonstrated below,

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().CircularGauge("circular2").CenterX("20").CenterY("20").Axes(axis  
=> axis.LineStyle(ls =>  
ls.Color("#F8F8F8").Width(2)).StartAngle(0).EndAngle(180).Add()).Render()
```

POSITION.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

In Percentage

By setting the value in percentage, gauge gets its mid point with respect to its plot area. For example, when the [centerX](#) value as '0%' and [centerY](#) value is '50%', gauge will get positioned at the top left corner of the plot area.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").CenterX("10%").CenterY("50%").Axes(ax
is => axis.LineStyle(ls =>
ls.Color("#F8F8F8").Width(2)).StartAngle(0).EndAngle(180).Add()).Render()
```

POSITION-PERCENTAGE.CS

```
using Microsoft.AspNetCore.Mvc;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Area Customization

Customize the gauge background

Using [background](#) and [border](#) properties, you can change the background color and border of the circular gauge.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Background("skyblue").Border(border
=>border.Width(2).Color("#FF0000")).Axes(axis => axis
```

```
.Radius("90 %").Maximum(120).StartAngle(230).EndAngle(130).LineStyle(ls
=>ls.Width(2)).MajorTicks(mt =>mt.Width(1).Color("#8c8c8c"))
.MinorTicks(mi => mi.Color("#8c8c8c").Width(1)).Add()).Render()
```

AREA-CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.CircularGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<CircularGaugePointer> pointers = new
List<CircularGaugePointer>();
            CircularGaugePointer pointer1 = new CircularGaugePointer();
            pointer1.Value = 60;
            pointer1.Radius = "60%";
            pointers.Add(pointer1);
            ViewBag.pointers = pointers;
            List<CircularGaugeRange> ranges = new
List<CircularGaugeRange>();
            CircularGaugeRange range1 = new CircularGaugeRange();
            range1.Start = 0;
            range1.End = 70;
            range1.Radius = "110%";
            range1.StartWidth = "10";
            ranges.Add(range1);
            CircularGaugeRange range2 = new CircularGaugeRange();
            range2.Start = 70;
            range2.End = 110;
            range2.Radius = "110%";
            range2.StartWidth = "10";
            ranges.Add(range2);
            CircularGaugeRange range3 = new CircularGaugeRange();
            range3.Start = 110;
            range3.End = 120;
            range3.Radius = "110%";
            range3.StartWidth = "10";
            ranges.Add(range3);
            ViewBag.ranges = ranges;
            return View();
        }
    }
}
```

Gauge Margin

You can set margin for gauge from its container through [margin](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular").Margin(margin => margin
.Bottom(40).Left(40).Right(40).Top(40)).Background("skyblue").Border(border
=>border.Width(2).Color("#FF0000")).Axes(axis => axis
.Radius("90 %").Maximum(120).StartAngle(230).EndAngle(130).LineStyle(ls
=>ls.Width(2)).MajorTicks(mt =>mt.Width(1).Color("#8c8c8c"))
.MinorTicks(mi => mi.Color("#8c8c8c").Width(1)).Add()).Render()
```

MARGIN.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.CircularGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<CircularGaugePointer> pointers = new
List<CircularGaugePointer>();
            CircularGaugePointer pointer1 = new CircularGaugePointer();
            pointer1.Value = 60;
            pointer1.Radius = "60%";
            pointers.Add(pointer1);
            ViewBag.pointers = pointers;
            List<CircularGaugeRange> ranges = new
List<CircularGaugeRange>();
            CircularGaugeRange range1 = new CircularGaugeRange();
            range1.Start = 0;
            range1.End = 70;
            range1.Radius = "110%";
            range1.StartWidth = "10";
            ranges.Add(range1);
            CircularGaugeRange range2 = new CircularGaugeRange();
            range2.Start = 70;
            range2.End = 110;
            range2.Radius = "110%";
            range2.StartWidth = "10";
            ranges.Add(range2);
            CircularGaugeRange range3 = new CircularGaugeRange();
            range3.Start = 110;
            range3.End = 120;
            range3.Radius = "110%";
            range3.StartWidth = "10";
            ranges.Add(range3);
            ViewBag.ranges = ranges;
```

```

        return View();
    }
}

```

Radius calculation based on angles

Render semi or quarter circular gauges by modifying the start and end angles. By enabling the radius based on angle option, the radius of circular gauge will be calculated based on the start and end angles to avoid excess white space.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().CircularGauge("circular2").MoveToCenter(true).Axes(axis =>
axis.LineStyle(ls =>
ls.Color("#F8F8F8").Width(2)).StartAngle(0).EndAngle(180).Add()).Render()

```

POSITION-PERCENTAGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

Accessibility in Circular Gauge component

Circular Gauge has built-in accessibility features like screen reading and WAI-ARIA attributes.

WAI-ARIA attributes

The Circular Gauge component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Circular Gauge component:

| Attributes | Purpose |

| --- | --- |

| **role=region** | It is specified in the pointer where the interactive drag and drop function is supported to update the pointer value. |

| **aria-label** | Provides an accessible name for the axis labels, legend title, legend item label, text pointer and annotation. |

Screen reading in Circular Gauge

Accessibility in the Circular Gauge component ensures that all users, regardless of ability or disability, can use screen reading. The following Circular Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Axis labels | Reads the axis labels of the Circular Gauge. |

| Legend title | Reads the title of the legend in the Circular Gauge. |

| Legend item label | Reads the label of the legend item in the Circular Gauge. |

| Text pointer | Reads the text content shown as a pointer in Circular Gauge. |

| Annotation | Reads the content specified in the annotation. |

Ensuring accessibility

The Circular Gauge component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Circular Gauge component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Circular Gauge component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Internationalization in Circular Gauge Control

Circular Gauge provides internationalization support for below elements.

- Axis Labels
- Tooltip

Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales.

Internationalization library is used to globalize number in CircularGauge component using [Format](#) property in [LabelStyle](#).

<!-- markdownlint-disable MD036 -->

Numeric Format

In the below example axis labels are globalized to **EUR**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
```

```
@Html.EJS().CircularGauge("circular").Locale("de").Axes(axis => axis
.LabelStyle(ls => ls.Position(Position.Inside).Format("c")).Add()).Render()
<script>
    ej.base.setCulture('de');
    ej.base.setCurrencyCode('EUR');
</script>
```

INTERNATIONALIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Right-to-left

Circular Gauge can render its elements from right to left, which improves the user experience for certain language users. To do so, set the [EnableRtl](#) property to **true**. When this property is enabled, elements such as the tooltip and legend will be rendered from right to left. Meanwhile, the axis can be rendered from right to left by setting the [Direction](#) property to **AntiClockWise**. For more information on axis, click [here](#).

The following example illustrates the right to left rendering of the Circular Gauge.

CSHTML

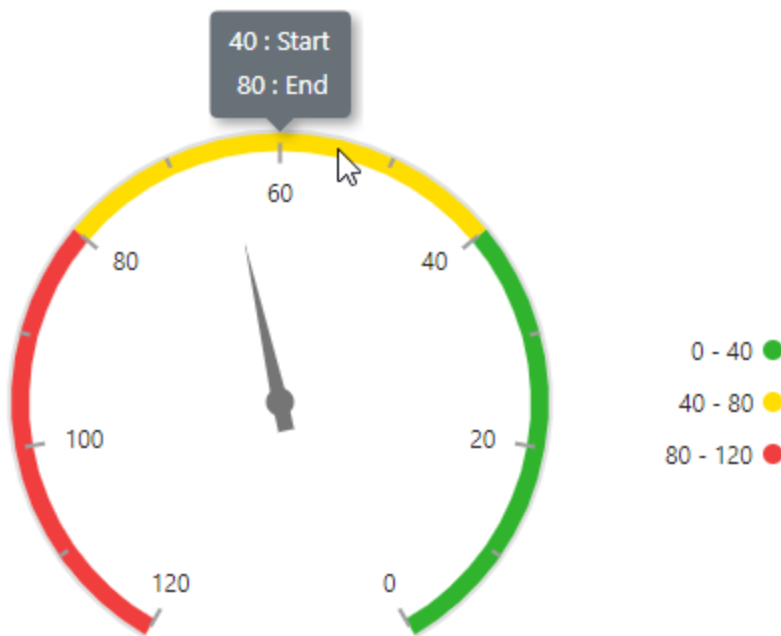
```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.CircularGauge;
@Html.EJS().CircularGauge("circular").EnableRtl(true).LegendSettings(legend
=> legend.Visible(true)).Tooltip(tooltip => tooltip.Enable(true).Type(new
string[] { "Pointer", "Range" }).EnableAnimation(false).Format("Pointer :
{value} ")).Axes(axis =>
axis.StartAngle(210).EndAngle(150).Minimum(0).Maximum(120).Radius("80%").Dir
ection(Syncfusion.EJ2.CircularGauge.GaugeDirection.AntiClockWise)
.LineStyle(ls => ls.Color("transparent").Width(10))
.MajorTicks(majortick =>
majortick.Offset(5).Color("#9E9E9E").Height(10)).MinorTicks(minortick =>
minortick.Height(0))
.LabelStyle(ls =>
ls.Position(Position.Inside).UseRangeColor(false)).Ranges(range =>
{
    range.Start(0).End(40).Color("#30B32D").Add();
}
```

```
        range.Start(40).End(80).Color("#FFDD00").Add();
        range.Start(80).End(120).Color("#F03E3E").Add();
    })
    .Pointers(pointer =>
    {
        pointer.Value(65).Color("#757575")

    .Radius("60%").MarkerShape(GaugeShape.Triangle).PointerWidth(8).NeedleTail(n
t => nt.Length("18%")).Cap(cap => cap.Radius(7).Color("#757575")).Add();
    }).Add()).Render()
```

RTL.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Migration from Essential JS 1

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Circular gauge dimensions

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Height** | **Property:** `height`

 `@Html.EJ().CircularGauge("container")`
 `.Height("400px")` | **Property:** `height`

 `@Html.EJS().CircularGauge("container")`
 `.Height("400px").Render()` |

| **Width** | **Property:** `width`

 `@Html.EJ().CircularGauge("container")`
 `.Width("100px")` | **Property:** `width`

 `@Html.EJS().CircularGauge("container")`
 `.Height("150px").Render()` |

| **Height(In Percentage)** | Not Applicable | **Property:** `height`

 `@Html.EJS().CircularGauge("container")`
 `.Height("50%").Render()` |

| **Width(In Percentage)** | Not Applicable | **Property:** `width`

 `@Html.EJS().CircularGauge("container")`
 `.Height("80%").Render()` |

Axis Line

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Axisline Width** | **Property:** `scales.size`

 `@Html.EJ().CircularGauge("container")`
 `.Scales(sc => sc.ShowScaleBar(true)`
 `.size(1).Add())` | **Property:** `axes.lineStyle.width`

 |

```
@Html.EJS().CircularGauge("container") <br/> .Axes(axes => axes.LineStyle(new  
CircularGaugeLine <br/> { Width = 0 }).Add()).Render()|
```

```
| Axisline Color| Property: scales.color<br/><br/> @Html.EJ().CircularGauge("container") <br/>  
.Scales(sc => sc.ShowScaleBar(true) <br/> .color('red').Add())| Property:  
axes.lineStyle.color<br/><br/> @Html.EJS().CircularGauge("container") <br/> .Axes(axes =>  
axes.LineStyle(new CircularGaugeLine <br/> { color = 'red' }).Add()).Render()|
```

```
| Axisline BackgroundColor| Not Applicable| Property: axes.background<br/><br/>  
@Html.EJS().CircularGauge("container") <br/> .Axes(axes => axes.Background("red") <br/>  
.Add()).Render()|
```

```
| Axisline Direction| Property: scales.direction<br/><br/> @Html.EJ().CircularGauge("container")  
<br/> .Scales(sc => sc.Direction('CounterClockwise') <br/> .color('red').Add())| Property:  
axes.direction<br/><br/> @Html.EJS().CircularGauge("container") <br/> .Axes(axes =>  
axes.Direction(GaugeDirection.AntiClockWise) <br/> .Add()).Render()|
```

```
| Axisline Radius| Property: scales.radius<br/><br/> @Html.EJ().CircularGauge("container") <br/>  
.Scales(sc => sc.Radius('150').color('red').Add())| Property: axes.radius<br/><br/>  
@Html.EJS().CircularGauge("container") <br/> .Axes(axes => axes.Radius(150) <br/>  
.Add()).Render()|
```

```
| Axisline Startangle| Property: scales.startAngle<br/><br/> @Html.EJ().CircularGauge("container")  
<br/> .Scales(sc => sc.startAngle('150') <br/> .color('red').Add())| Property:  
axes.startAngle<br/><br/> @Html.EJS().CircularGauge("container") <br/> .Axes(axes =>  
axes.StartAngle(150) <br/> .Add()).Render()|
```

```
| Axisline Endangle| Property: scales.sweepAngle<br/><br/> @Html.EJ().CircularGauge("container")  
<br/> .Scales(sc => sc.sweepAngle('150').color('red') <br/> .Add())| Property:  
axes.endAngle<br/><br/> @Html.EJS().CircularGauge("container").Axes(axes =>  
axes.EndAngle(70).Add()).Render()|
```

```
| Minimum Axisvalue| Property: scales.minimum<br/><br/> @Html.EJ().CircularGauge("container")  
<br/> .Scales(sc => sc.Minimum(20) <br/> .Add())| Property: axes.minimum<br/><br/>  
@Html.EJS().CircularGauge("container") <br/> .Axes(axes => axes.Minimum(20) <br/>  
.Add()).Render()|
```

```
| Maximum Axisvalue| Property: scales.maximum<br/><br/> @Html.EJ().CircularGauge("container")  
<br/> .Scales(sc => sc.maximum(200).Add())| Property: axes.maximum<br/><br/>  
@Html.EJS().CircularGauge("container") <br/> .Axes(axes => axes.Maximum(200) <br/>  
.Add()).Render()|
```

Ticks

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
| Type of Ticks| Property: scales.ticks.type<br/><br/> @Html.EJ().CircularGauge("container") <br/>  
.Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Major) <br/> .Add()}).Add()})| Property:
```

axes.majorTicks

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MajorTicks.Add())
 .Render()|

|Height of Major Ticks| **Property:** *scales.ticks.height*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Major)
 .Height(12).Add()).Add()})| **Property:** *axes.majorTicks.height*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MajorTicks(mi => mi.Height(12))
 .Add()).Render()|

|Width of Major Ticks| **Property:** *scales.ticks.width*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Major)
 .Width(3).Add()).Add()})| **Property:** *axes.majorTicks.width*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MajorTicks(mi => mi.Width(2))
 .Add()).Render()|

|Color of Major Ticks| **Property:** *scales.ticks.color*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Major)
 .color('red').Add()).Add()})| **Property:** *axes.majorTicks.color*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MajorTicks(mi => mi.Color("red"))
 .Add()).Render()|

|Offset of Major Ticks| **Property:** *scales.ticks.distanceFromScale*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Major)
 .DistanceFromScale(10).Add()).Add()})| **Property:** *axes.majorTicks.offset*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MajorTicks(mi => mi.Offset(5))
 .Add()).Render()|

|Angle of Major Ticks| **Property:** *scales.ticks.angle*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Major)
 .Angle(10).Add()).Add()})| Not Applicable|

|Interval of Major Ticks| **Property:** *scales.majorIntervalValue*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {MajorIntervalValue(10)
 .Add()})| **Property:** *axes.majorTicks.interval*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MajorTicks(mi => mi.Interval(2))
 .Add()).Render()|

|Height of Minor Ticks| **Property:** *scales.ticks.height*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Minor)
 .Height(12).Add()).Add()})| **Property:** *axes.minorTicks.height*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Height(12))
 .Add()).Render()|

|Width of Minor Ticks| **Property:** *scales.ticks.width*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Minor)
 .Width(3).Add()).Add()})| **Property:** *axes.minorTicks.width*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Width(2))
 .Add()).Render()|

|Color of Minor Ticks| **Property:** *scales.ticks.color*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic => {tic.Type(CircularTickTypes.Minor)

`.color('red').Add()).Add())| Property: axes.minorTicks.color

@Html.EJS().CircularGauge("container")
 .Axes(axes => axes.MinorTicks(mi =>
mi.Color("red"))
 .Add()).Render()|`

| Offset of Minor Ticks| **Property:** *scales.ticks.distanceFromScale*

@Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic =>
{tic.Type(CircularTickTypes.Minor)
 .DistanceFromScale(10).Add()}).Add()})| **Property:**
axes.minorTicks.offset

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
axes.MinorTicks(mi => mi.Offset(5))
 .Add()).Render()|

| Angle of Major Ticks| **Property:** *scales.ticks.angle*

@Html.EJ().CircularGauge("container")
 .Scales(sc => {Ticks(tic =>
{tic.Type(CircularTickTypes.Minor)
 .Angle(10).Add()}).Add()})| Not Applicable|

| Interval of Minor Ticks| **Property:** *scales.MinorIntervalValue*

@Html.EJ().CircularGauge("container")
 .Scales(sc => {MinorIntervalValue(10)

.Add()})| **Property:** *axes.minorTicks.interval*

 @Html.EJS().CircularGauge("container")

 .Axes(axes => axes.MinorTicks(mi => mi.Interval(2))
 .Add()).Render()|

Labels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Autoangle| **Property:** *scales.labels.autoAngle*

 @Html.EJ().CircularGauge("container")

 .Scales(sc => {Labels(lbl => {lbl.showLabels(true)
 .Add()}).Add()})| **Property:**
axes.labelStyle.autoAngle

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
axes.LabelStyle(ls => ls.AutoAngle(true))
 .Add()).Render()|

| Angle| **Property:** *scales.labels.angle*

 @Html.EJ().CircularGauge("container")

.Scales(sc => sc.showLabels(true).Labels(lbl => {lbl.Angle(20)
 .Add()}).Add()})| Not
Applicable|

| Offset| **Property:** *scales.labels.distanceFromScales*

@Html.EJ().CircularGauge("container")
 .Scales(sc => {Labels(lbl =>
{lbl.DistanceFromScale(10)
 .Add()}).Add()})| **Property:** *axes.labelStyle.offset*

@Html.EJS().CircularGauge("container")
 .Axes(axes => axes.LabelStyle(ls => ls.Offset(5))

 .Add()).Render()|

| Format| **Property:** *scales.labels.unitText*

 @Html.EJ().CircularGauge("container")

.Scales(sc => {Labels(lbl => {lbl.UnitText("Front")
 .Add()}).Add()})| **Property:**
axes.labelStyle.format

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
axes.LabelStyle(ls => ls.Format("KMPH"))
 .Add()).Render()|

| UnitText Position| **Property:** *scales.labels.placement*

@Html.EJ().CircularGauge("container")
 .Scales(sc => {Labels(lbl => {lbl.Placement("Near")

 .Add()}).Add()})| **Property:** *axes.labelStyle.position*

@Html.EJS().CircularGauge("container")
 .Axes(axes => axes.LabelStyle(ls =>
ls.Position(Position.Inside))
 .Add()).Render()|

|Label Range Color| Not Applicable| **Property:** *axes.labelStyle.useRangeColor*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.LabelStyle(ls =>
 ls.UseRangeColor(true))
 .Add()).Render()|

|LabelText Color| **Property:** *scales.labels.color*

 @Html.EJ().CircularGauge("container")

 .Scales(sc => {Labels(lbl => {lbl.Color("red")
 .Add()).Add()})| **Property:**
axes.labelStyle.font.color

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
 axes.LabelStyle(ls => ls.Font(fo => fo.Color("red")))
 .Add()).Render()|

|Opacity| **Property:** *scales.labels.opacity*

 @Html.EJ().CircularGauge("container")

 .Scales(sc => {Labels(lbl => {lbl.Opacity(0.2)
 .Add()).Add()})| **Property:**
axes.labelStyle.font.opacity

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
 axes.LabelStyle(ls => ls.Font(fo => fo.Opacity(0.5)))
 .Add()).Render()|

|Label Font Family| **Property:** *scales.labels.font.fontFamily*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Labels(lbl => {lbl.font('font').Add()})

 .Add()})
 var font = new {fontFamily="aerial"}| **Property:**
axes.labelStyle.font.fontFamily

 @Html.EJS().CircularGauge("container")
 .Axes(axes
 => axes.LabelStyle(ls => ls.Font(fo => fo.FontFamily("Roboto")))
 .Add()).Render()|

|Label Font Style| **Property:** *scales.labels.font.fontStyle*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Labels(lbl => {lbl.font('font')

 .Add()).Add()})
var font = new {font-style="bold"}| **Property:**
axes.labelStyle.font.fontStyle

 @Html.EJS().CircularGauge("container")
 .Axes(axes
 => axes.LabelStyle(ls => ls.Font(fo => fo.FontStyle("Bold")))
 .Add()).Render()|

|Label Font Size| **Property:** *scales.labels.font.size*

 @Html.EJ().CircularGauge("container")

 .Scales(sc => {Labels(lbl => {lbl.font('font').Add()})
 .Add()})
var font = new
 {size="12"}| **Property:** *axes.labelStyle.font.size*

 @Html.EJS().CircularGauge("container")

 .Axes(axes => axes.LabelStyle(ls => ls.Font(fo => fo.Size("12")))
 .Add()).Render()|

|Label Font Weight| Not Applicable| **Property:** *axes.labelStyle.font.fontWeight*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.LabelStyle(ls => ls.Font(fo =>
 fo
 .FontWeight("Regular")))
 .Add()).Render()|

Ranges

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Start Value| **Property:** *scales.ranges.startValue*

 @Html.EJ().CircularGauge("container")

 .Scales(sc => {Ranges(ran => {ran.StartValue(20)
 .Add()).Add()})| **Property:**
axes.ranges.start

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
 axes.Ranges(ran => ran.Start(20))
 .Add()).Render()|

|End Value| **Property:** *scales.ranges.endValue*

 @Html.EJ().CircularGauge("container")

 .Scales(sc => {Ranges(ran => {ran.EndValue(30)
 .Add()).Add()})| **Property:**
axes.ranges.end

 @Html.EJS().CircularGauge("container")
 .Axes(axes =>
 axes.Ranges(ran => ran.End(30))
 .Add()).Render()|

|Start Width| **Property:** *scales.ranges.startWidth*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.StartWidth(10)
 .Add()}).Add()})| **Property:** *axes.ranges.startWidth*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.StartWidth(10))
 .Add()).Render()|

|End Width| **Property:** *scales.ranges.endWidth*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.EndWidth(10)
 .Add()}).Add()})| **Property:** *axes.ranges.endWidth*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.EndWidth(10))
 .Add()).Render()|

|Color| **Property:** *scales.ranges.backgroundColor*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.BackgroundColor('red')
 .Add()}).Add()})| **Property:** *axes.ranges.color*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.Color("red"))
 .Add()).Render()|

|Offset| **Property:** *scales.ranges.distanceFromScale*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.DistanceFromScale(10))
 .Add()}).Add()})| Not Applicable|

|Placement| **Property:** *scales.ranges.placement*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.Placement('Far')
 .Add()}).Add()})| Not Applicable|

|Opacity| **Property:** *scales.ranges.opacity*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.Opacity(0.5)
 .Add()}).Add()})| Not Applicable|

|Radius| Not Applicable| **Property:** *axes.ranges.radius*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.Radius("30"))
 .Add()).Render()|

|Rounded Corner Radius| Not Applicable| **Property:** *axes.ranges.roundedCornerRadius*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.RoundedCornerRadius(10)).Add())
 .Render()|

|Border| **Property:** *scales.ranges.border*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Ranges(ran => {ran.Border.Color('blue')
 .Add()}).Add()})| Not Applicable|

Needle Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Needle Pointer| **Property:** *scales.pointers.type*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Type(PointerType.Needle)
 .Add()}).Add()})|

Property: *axes.pointers.type*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Needle)
 .Value(20)).Add()).Render()|

|Needle Pointer Color| **Property:** *scales.pointers.backgroundColor*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.BackgroundColor('red')
 .Add()}).Add()})| **Property:** *axes.pointers.color*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Color("red"))
 .Add()).Render()|

| Animation | **Property:** *enableAnimation*

 @Html.EJ().CircularGauge("container")
 .EnableAnimation(true) | **Property:** *axes.pointers.animation*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Animation(true)))
 .Add()).Render() |

| Pointer Width | **Property:** *scales.pointers.width*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Width(10)
 .Add()).Add()}) | **Property:** *axes.pointers.pointerWidth*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.PointerWidth(10)))
 .Add()).Render() |

| Pointer Radius | **Property:** *scales.pointers.distanceFromScale*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.DistanceFromScale(10)
 .Add()).Add()}) | **Property:** *axes.pointers.radius*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Radius("10")))
 .Add()).Render() |

| Opacity | **Property:** *scales.pointers.opacity*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Opacity(0.5)
 .Add()).Add()}) | Not Applicable |

| Needle Type | **Property:** *scales.pointers.needleType*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.NeedleType("Triangle")
 .Add()).Add()}) | Not Applicable |

| Back Needle Length | **Property:** *scales.pointers.backNeedleLength*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.BackNeedleLength(10)
 .Add()).Add()}) | **Property:** *axes.pointers.needleTail.length*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.NeedleTail(nd => nd.Length("10")))
 .Add()).Render() |

Marker Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Pointer | **Property:** *scales.pointers.type*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Type(PointerType.Marker)
 .Add()).Add()}) | **Property:** *axes.pointers.type*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Marker)
 .Value(20)).Add()).Render() |

| Marker Type | **Property:** *scales.pointers.markerType*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.MarkerType(MarkerType.Diamond)
 .Add()).Add()}) | **Property:** *axes.pointers.markerShape*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Marker)
 .MarkerShape(GaugeShape.Diamond)).Add()).Render() |

| Marker Width | **Property:** *scales.pointers.width*

 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Width(10)
 .Add()).Add()}) | **Property:** *axes.pointers.markerWidth*

 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Marker)
 .MarkerWidth(10)).Add()).Render() |

| Marker Height | **Property:** *scales.pointers.length*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Length(20) .Add()}).Add()}) | **Property:**
axes.pointers.markerHeight
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Marker) .MarkerHeight(20)).Add()).Render()

| Marker Image | **Property:** *scales.pointers.imageUrl*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.ImageUrl('a.png') .Add()}).Add()}) | **Property:**
axes.pointers.imageUrl
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Marker) .ImageUrl("a.png")).Add()).Render()

| Border Customization | **Property:** *scales.pointers.border*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {Pointers(po => {po.Border.Color('red') .Add()}).Add()}) | **Property:**
axes.pointers.border
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.Marker).Border(bo => bo.Color("red") .Width(2))).Add()).Render()

Rangebar Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Rangebar | Not Applicable | **Property:** *axes.pointers.type*
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.RangeBar) .Value(20)).Add()).Render()

| Rounded Corner Radius | Not Applicable | **Property:** *axes.pointers.roundedCornerRadius*
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Pointers(pt => pt.Type(PointerType.RangeBar) .RoundedCornerRadius(10)).Add()).Render()

Annotations

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Content | **Property:** *scales.customLabels.value*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {CustomLabels(c1 => {c1.Value("Circular Gauge") .Add()}).Add()}) | **Property:**
axes.annotations.content
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an => an.Content("Circular Gauge")) .Add()).Render()

| Angle | **Property:** *scales.customLabels.textAngle*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {CustomLabels(c1 => {c1.TextAngle(90) .Add()}).Add()}) | **Property:**
axes.annotations.angle
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an => an.Angle(90)) .Add()).Render()

| Font Family | **Property:** *scales.customLabels.font.fontFamily*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {CustomLabels(c1 => {c1.Font(fon => fon .FontFamily("Arial") .Add()}).Add()}) | **Property:**
axes.annotations.textStyle.fontFamily
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an => an.TextStyle(text => text.FontFamily("Roboto"))) .Add()).Render()

| Font Color | **Property:** *scales.customLabels.color*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {CustomLabels(c1 => {c1.Color('red')
 .Add()}).Add()}) | **Property:**
axes.annotations.textStyle.color
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an => an.TextStyle(text => text.Color("Red"))
 .Add()).Render() |

| Auto Angle | Not Applicable | **Property:** *axes.annotations.autoAngle*
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an =>
 an.AutoAngle(true))
 .Add()).Render() |

| Radius | Not Applicable | **Property:** *axes.annotations.radius*
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an =>
 an.Radius("10"))
 .Add()).Render() |

| Annotation Position | **Property:** *scales.customLabels.position*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {CustomLabels(c1 =>
 {c1.Position(pos => pos.X(5)
 .Y(-10).Add()).Add()}).Add()}) | Not Applicable |

| Annotation Position Type | **Property:** *scales.customLabels.positionType*
 @Html.EJ().CircularGauge("container")
 .Scales(sc => {CustomLabels(c1 =>
 {c1.CustomLabelPositionType
 (CustomLabelPositionType.Outer)
 .Add()}).Add()}) | Not
 Applicable |

| ZIndex | Not Applicable | **Property:** *axes.annotations.zIndex*
 @Html.EJS().CircularGauge("container")
 .Axes(axes => axes.Annotations(an =>
 an.ZIndex("1"))
 .Add()).Render() |

Appearance

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Title | Not Applicable | **Property:** *title*
 @Html.EJS().CircularGauge("container")
 .Title("Circular Gauge").Render() |

| Background Color | **Property:** *backgroundColor*
 @Html.EJ().CircularGauge("container")
 .BackgroundColor('red') | **Property:** *background*
 @Html.EJS().CircularGauge("container")
 .Background("Red").Render() |

| Localization | **Property:** *locale*
 @Html.EJ().CircularGauge("container")
 .Locale('en-US') | **Property:** *locale*
 @Html.EJS().CircularGauge("container")
 .Locale("en-US").Render() |

| Border | Not Applicable | **Property:** *border*
 @Html.EJS().CircularGauge("container")
 .Border(bo => bo.Color("red").Width(2)).Render() |

| Center of X | Not Applicable | **Property:** *centerX*
 @Html.EJS().CircularGauge("container")
 .CenterX("120px").Render() |

| Center of Y | Not Applicable | **Property:** *centerY*
 @Html.EJS().CircularGauge("container")
 .CenterY("150px").Render() |

| Theme | **Property:** *theme*

 @Html.EJ().CircularGauge("container")

 .Theme('Material') | **Property:** *theme*

 @Html.EJS().CircularGauge("container")

 .Theme(GaugeTheme.Material).Render() |

| Margin | Not Applicable | **Property:** *margin*

 @Html.EJS().CircularGauge("container")

 .Margin(mar => mar.Left(10).Right(10)).Render() |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Annotation Event | **Event:** *drawCustomLabel*

 @Html.EJ().CircularGauge("container")

 .DrawCustomLabel("customLabel")
 function customLabel(args) {} | **Event:**
annotationRender

 @Html.EJS().CircularGauge("container")

 .AnnotationRender("annotationRender").Render()
 <script>function
 annotationRender(args) {}</script> |

| Label Event | **Event:** *drawLabels*

 @Html.EJ().CircularGauge("container")

 .DrawLabels("drawLabels")
 function drawLabels(args) {} | **Event:** *axisLabelRender*

 @Html.EJS().CircularGauge("container")
 .AxisLabelRender("axisLabelRender").Render()

 <script>function axisLabelRender(args) {}</script> |

| Load Event | **Event:** *load*

 @Html.EJ().CircularGauge("container")
 .Load("load")

 function load(args) {} | **Event:** *load*

 @Html.EJS().CircularGauge("container")

 .Load("load").Render()
 <script>function load(args) {}</script> |

| Loaded Event | **Event:** *loaded*

 @Html.EJ().CircularGauge("container")

 .Loaded("loaded")
 function loaded(args) {} | **Event:** *loaded*

 @Html.EJS().CircularGauge("container")
 .Loaded("loaded").Render()

 <script>function loaded(args) {}</script> |

| Tooltip Rendered Event | Not Applicable | **Event:** *tooltipRender*

 @Html.EJS().CircularGauge("container")
 .TooltipRender("tooltipRender").Render()

 <script>function tooltipRender(args) {}</script> |

| Resized Rendered Event | Not Applicable | **Event:** *resized*

 @Html.EJS().CircularGauge("container")
 .Resized("resized").Render()
 function
 resized(args) {} |

| Animation Event | Not Applicable | **Event:** *animationComplete*

 @Html.EJS().CircularGauge("container")

 .AnimationComplete("animationComplete").Render()
 <script>function
 animationComplete(args) {}</script> |

| Mousedown Event | **Event:** *mouseClick*

 @Html.EJ().CircularGauge("container")

 .MouseClick("mouseClick")
 function mouseClick(args) {} | **Event:**
gaugeMouseDown

 @Html.EJS().CircularGauge("container")

 .GaugeMouseDown("gaugeMouseDown").Render()
 <script>function
 gaugeMouseDown(args) {}</script> |

| Mousemove Event | **Event:** *mouseClickMove*

 @Html.EJ().CircularGauge("container")

 .MouseClickedMove("mouseClickMove")

 function mouseClickedMove(args) {} | **Event:**
gaugeMouseLeave

 @Html.EJS().CircularGauge("container")

 .GaugeMouseLeave("gaugeMouseLeave").Render()
 <script>function
 gaugeMouseLeave(args) {}</script> |

| Mouseup Event | **Event:** *mouseClickUp*

 @Html.EJ().CircularGauge("container")

 .MouseClickedUp("mouseClickUp")

 function mouseClickedUp(args) {} | **Event:**
gaugeMouseUp

 @Html.EJS().CircularGauge("container")

 .GaugeMouseUp("gaugeMouseUp").Render()
 <script>function gaugeMouseUp(args)
 {}</script> |

| Pointerdrag Move Event | **Event:** *drawPointers*

 @Html.EJ().CircularGauge("container")

 .DrawPointers("drawPointers")

 function drawPointers(args) {} | **Event:**
dragMove

 @Html.EJS().CircularGauge("container")

 .DragMove("dragMove").Render()
 <script>function dragMove(args) {}</script> |

| Draw Range Event | **Event:** *drawRange*

 @Html.EJ().CircularGauge("container")

 .DrawRange("drawRange")

 function drawRange(args) {} | Not Applicable |

| Draw Ticks Event | **Event:** *drawTicks*

 @Html.EJ().CircularGauge("container")

 .DrawTicks("drawTicks")

 function drawTicks(args) {} | Not Applicable |

| Legend Render Event | **Event:** *legendItemRender*

 @Html.EJ().CircularGauge("container")

 .LegendItemRender("legendItemRender")

 function legendItemRender(args) {} | Not
 Applicable |

| Animation Complete Event | Not Applicable | **Event:** *animationComplete*

 @Html.EJS().CircularGauge("container")

 .AnimationComplete("animationComplete").Render()
 <script>function
 animationComplete(args) {}</script> |

| Right Click Event | **Event:** *rightClick*

 @Html.EJ().CircularGauge("container")

 .RightClick("rightClick")

 function rightClick(args) {} | Not Applicable |

| Double Click Event | **Event:** *doubleClick*

 @Html.EJ().CircularGauge("container")

 .DoubleClick("doubleClick")

 function doubleClick(args) {} | Not Applicable |

How To

How To

Get data from database for circular gauge and render gauge from code behind

Circular gauge can be rendered from the code behind by initializing the required properties in controller and passing them through ViewData to client-side. You can get the data from the database by creating a data table and accessing the data table values to gauge pointers value.

To get the data from data base and render the gauge from code behind, follow the given steps:

Step 1: Initialize the required properties in controller and get data from database, create a new data collection that contains minimum, maximum, and value attributes from the data base data, and then assign the data to the minimum, maximum, and pointer values of gauge.

```
`cs
public ActionResult Index()
{
    //Initialize the chart model
    CircularGauge gaugeModel = new CircularGauge();
    // Add the required properties for chart using the following method
    InitializeGauge(gaugeModel);
    // Store the chart model
    ViewData["GaugeModel"] = gaugeModel;
    return View();
}

private void InitializeGauge(CircularGauge gaugeModel)
{
    // Create a data base data table
    DataTable dt = new DataTable();
    dt.Columns.Add("Name");
    dt.Columns.Add("MaxStock");
    dt.Columns.Add("FreeStock");
    dt.Columns.Add("AvailableStock");
    dt.Rows.Add(new Object[] { "T13", 10000, 5000, 8230 });
    // Create a new data collection from data base table
    GaugeData gauges = new GaugeData();
    foreach (DataRow row in dt.Rows)
    {
        gauges.Minimum = Convert.ToDouble(row["FreeStock"]);
        gauges.Maximum = Convert.ToDouble(row["MaxStock"]);
        gauges.Value = Convert.ToDouble(row["AvailableStock"]);
    }

    CircularGaugePointer pointer = new CircularGaugePointer();
    pointer.Value = gauges.Value;
    CircularGaugeAxis axis = new CircularGaugeAxis();
    axis.Pointers = new List<CircularGaugePointer>();
    axis.Minimum = gauges.Minimum;
```

```
axis.Maximum = gauges.Maximum;
axis.Pointers.Add(pointer);
gaugeModel.Axes = new List<CircularGaugeAxis>();
gaugeModel.Axes.Add(axis);
}
,
```

Step 2: Bind the gauge properties passed via ViewData from the controller at client-side as demonstrated in the following code sample.

```
`html
```

```
@Html.EJS().CircularGauge("container",
(Syncfusion.EJ2.CircularGauge.CircularGauge)ViewData["GaugeModel"]).Render();
,
```

CSHTML

```
@Html.EJS().CircularGauge("container",
(Syncfusion.EJ2.CircularGauge.CircularGauge)ViewData["GaugeModel"]).Render()
;
```

DATABASE.CS

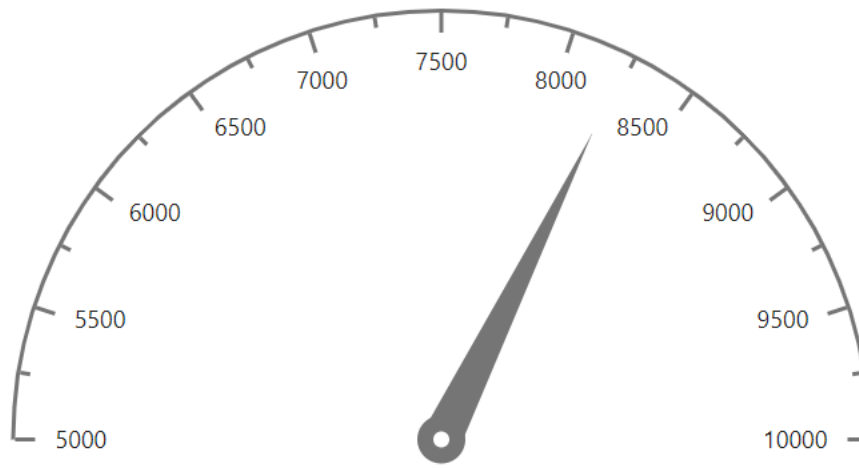
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            //Initialize the chart model
            CircularGauge gaugeModel = new CircularGauge();
            // Add the required properties for chart using the following
method
            InitializeGauge(gaugeModel);
            // Store the chart model
            ViewData["GaugeModel"] = gaugeModel;
            return View();
        }
        private void InitializeGauge(CircularGauge gaugeModel)
        {
            // Create a data base data table
            DataTable dt = new DataTable();
            dt.Columns.Add("Name");
            dt.Columns.Add("MaxStock");
```



```
dt.Columns.Add("FreeStock");
dt.Columns.Add("AvailableStock");
dt.Rows.Add(new Object[] { "T13", 10000, 5000, 8230 });
// Create a new data collection from data base table
GaugeData gauges = new GaugeData();
foreach (DataRow row in dt.Rows)
{
    gauges.Minimum = Convert.ToDouble(row["FreeStock"]);
    gauges.Maximum = Convert.ToDouble(row["MaxStock"]);
    gauges.Value = Convert.ToDouble(row["AvailableStock"]);
}
CircularGaugePointer pointer = new CircularGaugePointer();
pointer.Value = gauges.Value;
CircularGaugeAxis axis = new CircularGaugeAxis();
axis.Pointers = new List<CircularGaugePointer>();
axis.Minimum = gauges.Minimum;
axis.Maximum = gauges.Maximum;
axis.Pointers.Add(pointer);
gaugeModel.Axes = new List<CircularGaugeAxis>();
gaugeModel.Axes.Add(axis);
}
}
public class GaugeData
{
    public double Minimum;
    public double Maximum;
    public double Value;
}
}
```

Sample reference : [circulargauge sample](#).

Screenshot



ColorPicker

Getting Started with ASP.NET MVC ColorPicker Control

This section briefly explains about how to include [ASP.NET MVC ColorPicker](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5

NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

`

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

`

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC ColorPicker control

Now, add the Syncfusion ASP.NET MVC ColorPicker control in **~/Views/Home/Index.cshtml** page.

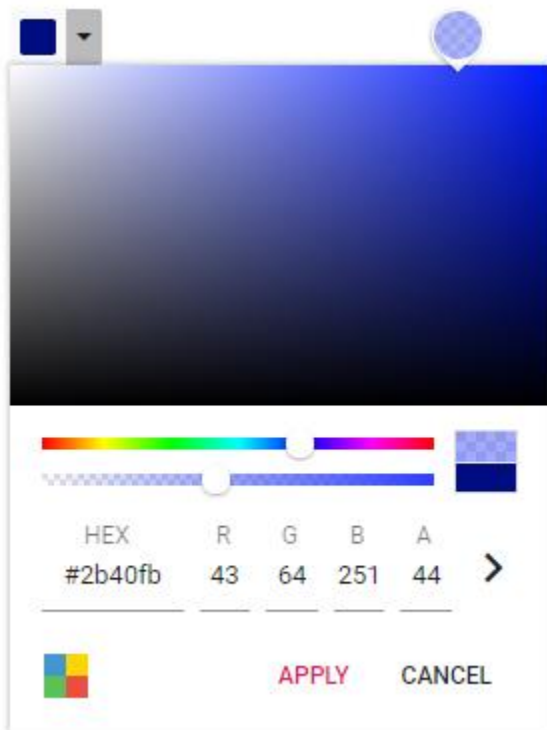
CSHTML

```

<div class='wrap'>
  <h4>Choose color</h4>
  @Html.EJS().ColorPicker("element").Render()
</div>
<style>
  .wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
  }
</style>

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC ColorPicker control will be rendered in the default web browser.



Inline type

By default, the ColorPicker will be rendered using SplitButton and open the pop-up to access the ColorPicker. To render the ColorPicker container alone and to access it directly, render it as inline. It can be achieved by setting the [Inline](#) property to true.

The following sample shows the inline type rendering of ColorPicker.

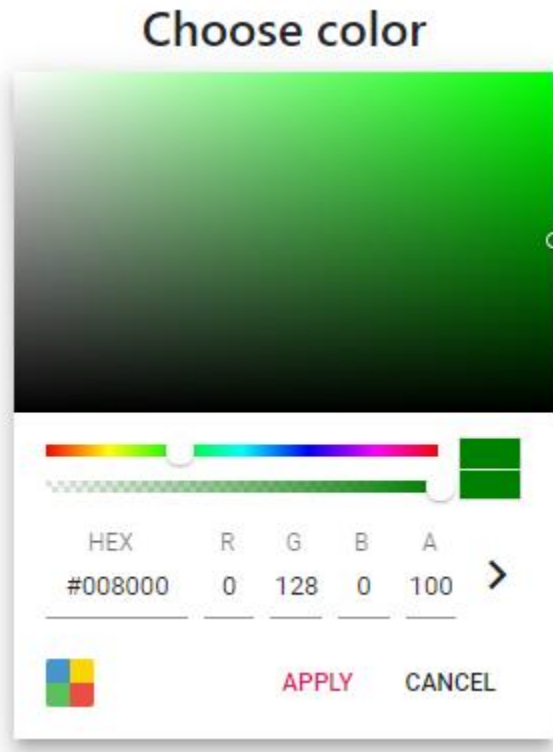
CSHTML

```

<div class='wrap'>
  <h4>Choose color</h4>
  @Html.EJS().ColorPicker("element").Inline(true).Render()
</div>
<style>
  .wrap {

```

```
margin: 0 auto;  
width: 300px;  
text-align: center;  
}  
</style>
```



Note: The [ShowButtons](#) property is disabled in this sample because the control buttons are not needed for inline type. To know about the control buttons functionality, refer to the [showButtons](#) document.

Note: [View Sample in GitHub](#).

See also

- [Set color value](#)
- [ColorPicker customization](#)

Mode and Value in Color Picker Control

Rendering palette at initial load

By default, the **Picker** area will be rendered at initial load. To render the **Palette** area while opening the ColorPicker pop-up, and specify the [Mode](#) property as **Palette**.

In the following sample, it will render the **Palette** at initial load.

CSHTML

```
<div class='wrap'>  
  <h4>Select Color</h4>
```

```
@Html.EJS().ColorPicker("element").Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette).Render()
</div>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
</style>
```

PALETTE.CS

```
public ActionResult Palette()
{
    return View();
}
```

Color value

The [Value](#) property can be used to specify the color value to the ColorPicker. It supports either **three** or **six** digit hex codes. To include **opacity**, set the color value as **four** or **eight** digit hex code.

In the following sample, the color value sets as **four** digit hex code, the last digit represents the **opacity** value.

CSHTML

```
<div class='wrap'>
    <h4>Choose Color</h4>

    @Html.EJS().ColorPicker("element").Value("035a").ModeSwitcher(false).Render()
</div>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
</style>
```

VALUE.CS

```
public ActionResult Value()
{
    return View();
}
```

Note: The [Value](#) property supports hex code with or without **#** prefix.

Note: [View Sample in GitHub.](#)

See also

- [How to render palette alone](#)
- [Custom palette](#)
- [No color support in palette](#)

Localization and RTL

Localization

The **Localization** library allows to localize default text content of the ColorPicker. The ColorPicker component has static text for control buttons (apply / cancel) and mode switcher that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the ColorPicker.

| Locale key words | Text |

| ----- | ----- |

| Apply | Apply |

| Cancel | Cancel |

| ModeSwitcher | Switch Mode |

Loading translations

To load translation object in an application use **load** function of **L10n** class.

The following example demonstrates the ColorPicker in **Deutsch** culture.

CSHTML

```
<div class='wrap'>
  <h4>Choose Color</h4>
  @Html.EJS().ColorPicker("element").Locale("de-DE").Render()
</div>
<script>
  ej.base.L10n.load({
    'de-DE': {
      'colorpicker': {
        'Apply': 'Anwenden',
        'Cancel': 'Abbrechen',
        'ModeSwitcher': 'Modus wechseln'
      }
    }
  });
</script>
<style>
  .wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
  }
</style>
```

LOCALIZATION.CS

```
public ActionResult Localization()  
{  
    return View();  
}
```

Right to Left - RTL

ColorPicker component has **RTL** support. It helps to render the ColorPicker from right-to-left direction. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc). This can be achieved by setting the [enableRtl](#) property to **true**.

CSHTML

```
<div class='wrap'>  
    <h4>Choose Color</h4>  
    @Html.EJS().ColorPicker("element").EnableRtl(true).Locale("ar-AE").Render()  
</div>  
<script>  
    ej.base.L10n.load({  
        'ar-AE': {  
            'colorpicker': {  
                'Apply': 'تطبيق',  
                'Cancel': 'إلغاء',  
                'ModeSwitcher': 'مفتاح كهربائي الوضع'  
            }  
        }  
    });  
</script>  
<style>  
    .wrap {  
        margin: 0 auto;  
        width: 300px;  
        text-align: center;  
    }  
</style>
```

RTL.CS

```
public ActionResult Rtl()  
{  
    return View();  
}
```

Note: [View Sample in GitHub](#).

See also

- [More information about localization](#)

Accessibility in Color Picker Control

The Color picker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Color picker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Color picker component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Color picker component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Color picker component as **color** and the tiles as **gridcell** in the color palette. |

| **aria-label** | Indicates the accessible name for the tiles. |

| **aria-selected** | Indicates the current selected state of the tile. |

| **aria-haspopup** | Indicates the availability of the popup element. |

| **aria-expanded** | Indicates whether the popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Color picker component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Color picker component.

| Press | To do this |

| --- | --- |

| Up Arrow | Moves the handler/tile up from the current position. |

| Down Arrow | Moves the handler/tile down from the current position. |

| Left Arrow | Moves the handler/tile left from the current position. |

| Right Arrow | Moves the handler/tile right from the current position. |

| Enter | Apply the selected color value. |

| Tab | To focus the next focusable element in the Color picker popup. |

Ensuring accessibility

The Color picker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Color picker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Color picker component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Styles and Appearances

To modify the ColorPicker appearance, you need to override the default CSS of ColorPicker component. Find the list of CSS classes and its corresponding section in ColorPicker component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ----- | ----- |

| .e-custom-picker .e-container .e-handler | To customized Color Picker selection handler |

| .color-picker.e-dropdown-popup ul .e-container | To customize the Color Picker container |

| .color-picker.e-dropdown-popup ul .e-item.e-palette-item | To customize the Color Picker palette item |

| .color-picker.e-dropdown-popup .e-container .e-switch | To customize the Color Picker switch control |

| .color-picker.e-dropdown-popup .e-container .e-slider-preview | To customize the Color Picker slider control |

How To

Hide control buttons in Color Picker Control

ColorPicker can be rendered without control buttons (Apply/Cancel). In this case, while selecting a color, the ColorPicker pop-up is closed and selected colors can be applied directly. To hide control buttons, set the [showButtons](#) property to `false`.

CSHTML

```
<div class='wrap'>
  <h4>Choose Color</h4>
  @Html.EJS().ColorPicker("element").ShowButtons(false).Render()
</div>
<style>
  .wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
  }
</style>
```

SHOW-BUTTONS.CS

```
public ActionResult ShowButtons()
{
    return View();
}
```

Note: [View Sample in GitHub](#).

Render palette alone in Color Picker Control

To render the `Palette` alone in ColorPicker, specify the [mode](#) property as `Palette`, and set the [modeSwitcher](#) property to `false`.

In the following sample, the [showButtons](#) property is disabled to hide the control buttons and it renders only the **Palette** area.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div class='wrap'>
    <h4>Choose Color</h4>

    @Html.EJS().ColorPicker("element").Mode(ColorPickerMode.Palette).ModeSwitcher(false).ShowButtons(false).Render()
</div>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
</style>
```

PALETTE-ALONE.CS

```
public ActionResult PaletteAlone()
{
    return View();
}
```

Note: To render **Picker** alone specify the [mode](#) property as 'Picker'.

Note: [View Sample in GitHub.](#)

ColorPicker in DropDownButton

This section explains about how to render the ColorPicker in DropDownButton. The [target](#) property of the DropDownButton helps to achieve this scenario. To know about the usage of **target** property, refer to [Popup templating](#) section.

In the below sample, the color picker is rendered as inline type by setting [inline](#) property as **true** and the rendered color picker wrapper is passed as a **target** to the DropDownButton to achieve the above scenario.

CSHTML

```
<div class='wrap'>

    @Html.EJS().ColorPicker("colorpicker").Inline(true).Change("onChange").Render()
    <h4>Choose color</h4>
    @Html.EJS().DropDownButton("ddb").Target(".e-colorpicker-wrapper").IconCss("e-dropdownbtn-preview").BeforeClose("onDdbBeforeClose").Open("onDdbOpen").Render()
</div>
<script>
    // Triggers while changing the colors.
    function onChange(args) {
```

```

        document.getElementById('ddb').children[0].style.backgroundColor =
args.currentValue.rgba;
        closePopup();
    }
    // Triggers after the ddb popup open.
    function onDdbOpen(args) {
        args.element.parentElement.querySelector('.e-
cancel').addEventListener('click', closePopup);
    }
    // Triggers before closing the ddb popup.
    function onDdbBeforeClose(args) {
        args.element.parentElement.querySelector('.e-
cancel').removeEventListener('click', closePopup);
    }
    // Function to close the ddb popup.
    function closePopup() {
        ej.base.getInstance(document.getElementById('ddb'),
ejs.splitbuttons.DropDownButton).toggle();
    }
</script>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
    /* DropDownButton preview customization */
    #ddb .e-btn-icon.e-dropdownbtn-preview {
        background-color: #008000;
        height: 18px;
        width: 18px;
        margin-top: 0;
    }
    #ddb {
        padding: 4px;
    }
    h4 {
        font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
        font-size: 14px;
    }
</style>

```

DROPDOWNBTN.CS

```

public ActionResult Dropdownbtn()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Customize ColorPicker Control

Custom palette

By default, the Palette will be rendered with default colors. To load custom colors in the palette, specify the colors in the [presetColors](#) property. To customize the color palette, add a custom class to palette tiles using [beforeTileRender](#) event.

The following sample demonstrates the above functionalities.

CSHTML

```
<div class='wrap'>
  <div id="preview"></div>
  <h4>Select color</h4>

@Html.EJS().ColorPicker("element").Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette).Inline(true).ModeSwitcher(false).ShowButtons(false).Columns(4).BeforeTileRender("tileRender").PresetColors(@ViewBag.customColors).Change("onChange").Render()
</div>
<script>
  // Triggers while selecting colors from palette.
  function onChange(args) {
    document.getElementById('preview').style.backgroundColor =
args.currentValue.hex;
  }
  // Triggers before rendering each palette tile.
  function tileRender(args) {
    args.element.classList.add("e-icons");
    args.element.classList.add("e-custom-tile");
  }
</script>
<style>
.e-container {
  background-color: transparent;
  border-color: transparent;
  box-shadow: none;
}
.e-container .e-palette .e-custom-tile {
  border: 0;
  color: #fff;
  height: 36px;
  font-size: 18px;
  width: 36px;
  line-height: 36px;
  border-radius: 50%;
  margin: 2px 5px;
}
.e-container .e-custom-palette.e-palette-group {
  height: 182px;
}
.e-container .e-palette .e-custom-tile.e-selected::before {
  content: '\e933';
}
.e-container .e-palette .e-custom-tile.e-selected {
  outline: none;
}
```

```
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}
#preview {
    background-color: #ba68c8;
    height: 50px;
    width: 100%;
}
</style>
```

CUSTOM-PALETTE.CS

```
public ActionResult CustomPalette()
{
    List<object> custom = new List<object>();
    custom.Add(new
    {
        Custom1 = new string[] { "#ef9a9a", "#e57373", "#ef5350",
            "#f44336", "#f48fb1", "#f06292", "#ec407a", "#e91e63",
            "#ce93d8",
            "#ba68c8", "#ab47bc", "#9c27b0", "#b39ddb", "#9575cd",
            "#7e57c2", "#673AB7" },
        Custom2 = new string[] { "#9FA8DA", "#7986CB", "#5C6BC0",
            "#3F51B5",
            "#90CAF9", "#64B5F6", "#42A5F5", "#2196F3", "#81D4FA",
            "#4FC3F7", "#29B6F6", "#03A9F4",
            "#80DEEA", "#4DD0E1", "#26C6DA", "#00BCD4" },
        Custom3 = new string[] { "#80CBC4", "#4DB6AC", "#26A69A",
            "#009688",
            "#A5D6A7", "#81C784", "#66BB6A", "#4CAF50", "#C5E1A5",
            "#AED581", "#9CCC65", "#8BC34A", "#E6EE9C",
            "#DCE775", "#D4E157", "#CDDC39" },
        Custom4 = new string[] { "#FFF59D", "#FFF176", "#FFEE58",
            "#FFEB3B",
            "#FFE082", "#FFD54F", "#FFCA28", "#FFC107", "#FFCC80",
            "#FFB74D", "#FFA726", "#FF9800", "#FFAB91",
            "#FF8A65", "#FF7043", "#FF5722" }
    });
    ViewBag.customColors = custom[0];
    return View();
}
```

Hide input area from picker

By default, the input area will be rendered in ColorPicker. To hide the input area from it, add `e-hide-value` class to ColorPicker using the `CssClass` property.

In the following sample, the ColorPicker is rendered without input area.

CSHTML

```
<div class='wrap'>
    <h4>Choose Color</h4>
    @Html.EJS().ColorPicker("element").CssClass("e-hide-
value").ModeSwitcher(false).Render()
```

```
</div>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
</style>
```

HIDE-INPUT.CS

```
public ActionResult HideInput()
{
    return View();
}
```

Custom handle

Color picker handle shape and UI can be customized. Here, the handle has been customized as `svg icon`. The same way you can customize the handle based on your requirement.

CSHTML

```
<div class='wrap'>
    <h4>Choose Color</h4>
    @Html.EJS().ColorPicker("element").Value("#344aae").CssClass("e-custom-
picker").ModeSwitcher(false).Open("onOpen").Render()
</div>
<script>
    // Triggers after the color picker popup open.
    function onOpen(args) {
        args.element.querySelector('.e-handler').classList.add('e-icons');
    }
</script>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
    /* To hide the handle balloon preview */
    .e-color-picker-tooltip.e-popup.e-popup-open {
        display: none;
    }
    /* Handle customization styles */
    .e-custom-picker .e-container .e-hsv-container .e-handler {
        background: transparent
url('data:image/svg+xml;base64,PD94bWwgdmVyc2lvdj0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz4KPCEtLSBHZW5lcmF0b3I6IEFkb2JlIElsbHVzdHJhdG9yIDIyLjEueUMCwgU1ZHIEV4cG9yd
CBQbHVnLUluc4gU1ZHIHFZlcnNpb246IDYumDAGQnVpbGQGMCkgIC0tPgo8c3ZnIHZlcmluc4gU1ZHI
jEuMSIGaWQ9IkxheWVyXzEiIHhtbG5zPSJodHRwOi8vd3d3LnclLm9yZy8yMDAwL3N2ZyIgeG1sb
nM6eGxpbnMs9Imh0dHA6Ly93d3d3LnclLm9yZy8yMDAwL3N2ZyIgeG1sbm9uZS4ueUMCwgU1ZHIEV4c
G9ydCBQbHVnLUluc4gU1ZHIHFZlcnNpb246IDYumDAGQnVpbGQGMCkgIC0tPgo8c3ZnIHZlcmluc4gU1
ZHIjeE2LDYqMTAsNiAxMcwWIDYsMCA2LDYqMCwWIDAsMTAqNiwxMCA2LDE2IDEwLDE2IDEwLDEwIDE2L
```



```
DEwIAkiLz4KPC9nPgo8cGF0aCBkPSJNMTAsNlYwSDZ2NkgwdjRoNnY2aDR2LTZoNlY2SDEweiBNM
TUsOUg5djZINlY5SDFWN2g2VjFoMnY2aDZWOXoiLz4KPC9zdmc+Cg==');
    font-size: 16px;
    height: 16px;
    line-height: 16px;
    margin-left: -8px;
    margin-top: -8px;
    border: none;
    box-shadow: none;
    width: 16px;
}
</style>
```

CUSTOM-HANDLE.CS

```
public ActionResult CustomHandle()
{
    return View();
}
```

Custom primary button

By default, the applied color will be updated in primary button of the color picker. You can customize that as **icon**.

In the following sample, the **picker** icon is added to primary button and using [change](#) event, the selected color will be updated in bottom portion of the icon.

CSHTML

```
<div class='wrap'>
    <h4>Chosse color</h4>
    @Html.EJS().ColorPicker("element").CssClass("e-custom-
icon").Change("onChange").Render()
</div>
<script>
    var iconPreview;
    window.onload = function () {
        iconPreview = document.querySelector('.e-custom-icon .e-selected-
color');
        iconPreview.classList.add("e-icons");
    }
    function onChange(args) {
        iconPreview.style.borderBottomColor = args.currentValue.rgba;
    }
</script>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
    /* Icon customization */
    .e-custom-icon.e-colorpicker-wrapper .e-split-btn-wrapper .e-split-btn
.e-selected-color {
        background: none;
    }
</style>
```

```

        border-bottom-style: solid;
        border-bottom-width: 3px;
        width: 14px;
        margin: 0px 2px;
        border-bottom-color: #008000;
    }
    .e-custom-icon.e-colorpicker-wrapper .e-split-btn-wrapper .e-split-btn
    .e-selected-color .e-split-preview {
        display: none;
    }
    .e-custom-icon.e-colorpicker-wrapper .e-split-btn-wrapper .e-split-btn
    .e-selected-color::before {
        content: '\e35c';
    }
}
</style>

```

ICON.CS

```

public ActionResult Icon()
{
    return View();
}

```

Note: The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icon to customize the primary button.

Display hex code in input

The color picker input element can be showcased in the place of primary button. The applied color hex code will be updated in the primary button input.

CSHTML

```

<div class='wrap'>
    <h4>Choose color</h4>
    @Html.EJS().ColorPicker("element").CssClass("e-custom-input").Render()
</div>
<script>
    window.onload = function () {
        var target = document.getElementById("element");
        ej.base.attributes(target, { "type": "text" });
        ej.base.addClass([target], "e-input");
        // Moving the ColorPicker input element before the secondary button.
        target.nextElementSibling.insertBefore(target,
        target.nextElementSibling.children[1]);
    }
</script>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
    /* Input element customization */
    .e-custom-input.e-colorpicker-wrapper .e-split-btn-wrapper .e-input {
        height: 16px;
    }

```

```

        margin: 0;
        opacity: 1;
        position: initial;
        width: 75px;
    }
    /* To hide primary button */
    .e-custom-input.e-colorpicker-wrapper .e-split-btn-wrapper .e-split-btn
{
    display: none;
}
    /* Secondary button customization */
    .e-custom-input.e-colorpicker-wrapper .e-split-btn-wrapper .e-btn.e-
dropdown-btn {
        background: transparent;
        border-color: transparent;
        border-bottom-color: rgba(0, 0, 0, 0.42);
    }
    .e-custom-input.e-colorpicker-wrapper .e-split-btn-wrapper .e-
input:focus+.e-btn.e-dropdown-btn {
        padding-bottom: 3px;
        border-bottom-width: 2px;
        border-bottom-color: #e3165b;
    }
    /* Rotating the dropdown arrow */
    .e-custom-input.e-colorpicker-wrapper .e-split-btn-wrapper .e-btn.e-
dropdown-btn .e-caret {
        transform: rotate(0deg);
        transition: transform 200ms ease-in-out;
    }
    .e-custom-input.e-colorpicker-wrapper .e-split-btn-wrapper .e-btn.e-
dropdown-btn.e-active .e-caret {
        transform: rotate(180deg);
    }
}
</style>

```

INPUT.CS

```

public ActionResult Input()
{
    return View();
}

```

Custom UI

The color picker UI can be customized in all possible ways. The following sample shows the excel like UI customization with the help of SplitButton and Dialog component. In that by clicking the more colors option from color palette, the dialog contains color picker will open.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div class="wrap">
    <ul id="target" tabindex="0">
        <li class="e-item e-palette-item">

```

```

@Html.EJS().ColorPicker("palette").Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette).Inline(true).ModeSwitcher(false).ShowButtons(false).Change("onPaletteChange").Render()
    </li>
    <li class="e-item" tabindex="-1">
        <span class="e-menu-icon e-morecolors"></span>
        More colors...
    </li>
</ul>
<div>

@Html.EJS().ColorPicker("picker").Inline(true).ModeSwitcher(false).Created("onPickerCreated").Change("onPickerChange").Render()
    <h4>Select color</h4>
    @Html.EJS().SplitButton("splitbtn").IconCss("e-icons e-font-icon").Target("#target").Open("onDdPopupOpen").Created("onSplitBtnCreated").BeforeClose("onBeforeDdPopupClose").Render()
    @Html.EJS().Dialog("picker-dialog").CssClass("e-dlg-picker").Created("onLoad").AnimationSettings(new DialogAnimationSettings() { Effect = DialogEffect.Zoom }).IsModal(true).Visible(false).OverlayClick("closePickerDlg").Content("<div class='dlgContent'></div>").Width("270px").Height("336px").Target(".wrap").Render()
    </div>
</div>
<script>
var splitBtnIcon, dlg, picker;
function onSplitBtnCreated() {
    splitBtnIcon = document.getElementById("splitbtn");
}
function onPickerCreated() {
    picker = document.getElementById("picker");
}
function onLoad() {
    dlg = document.getElementById("picker-dialog");
    picker.nextElementSibling.querySelector('.e-ctrl-btn .e-cancel').addEventListener('click', closePickerDlg);
    dlg.querySelector(".dlgContent").appendChild(picker.parentElement);
}
function onPickerChange(args) {
    onPaletteChange(args);
    closePickerDlg();
}
function onPaletteChange(args) {
    splitBtnIcon.querySelector('.e-font-icon').style.borderBottomColor = args.currentValue.rgba;
}
function onDdPopupOpen(args) {
    args.element.children[1].addEventListener('click', openPickerDlg);
}
function onBeforeDdPopupClose(args) {
    args.element.children[1].removeEventListener('click', openPickerDlg);
}
function openPickerDlg() {
    dlg.ej2_instances[0].show();
}

```

```
}
function closePickerDlg() {
    dlg.ej2_instances[0].hide();
}
</script>
<style>
    .wrap {
        margin: 0 auto;
        width: 100%;
        height: 100%;
        min-height: 350px;
        text-align: center;
    }
    .e-btn-icon.e-font-icon {
        border-bottom-style: solid;
        border-bottom-width: 3px;
    }
    .e-btn-icon.e-font-icon::before {
        content: '\e34c';
    }
    .e-colorpicker-wrapper.e-hide-palette {
        display: none;
    }
    .e-dropdown-popup ul .e-item:first-child.e-palette-item {
        height: auto;
        padding: 0;
    }
    .e-dlg-picker.e-dialog .e-dlg-content {
        padding: 0;
        background-color: transparent;
    }
    .e-dlg-picker.e-dialog {
        max-height: 336px !important;
        background-color: transparent;
    }
    .e-dropdown-popup ul .e-item .e-menu-icon.e-morecolors {
        height: 24px;
        margin-top: 6px;
        width: 24px;
        background-image: linear-gradient(to bottom, #fff 0, #000 100%);
        background-color: #0450c2;
        background-blend-mode: hard-light;
    }
</style>
```

EXCEL-UI.CS

```
public ActionResult ExcelUi()
{
    return View();
}
```

Note: [View Sample in GitHub.](#)

Handle no color support in Color Picker Control

The ColorPicker component supports no color functionality. By clicking the no color tile from palette, the selected color becomes **empty** and considered as no color has been selected from color picker.

Default no color

To achieve this, set `noColor` property as `true`.

In the following sample, the first tile of the color palette represents the no color tile. By clicking the no color tile you can achieve the above functionalities.

CSHTML

```
<div class='wrap'>
  <div id="preview"></div>
  <h4>Select color</h4>

@Html.EJS().ColorPicker("element").Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette).Created("onCreated").NoColor(true).Value("#ba68c8").ModeSwitcher(false).ShowButtons(false).Change("onChange").Render()
</div>
<script>
  var preview;
  function onCreated() {
    preview = document.getElementById("preview");
    // Initially set the default color and value to preview.
    preview.style.backgroundColor = "#ba68c8";
    preview.textContent = "#ba68c8";
  }
  // Triggers while selecting colors from palette.
  function onChange(args) {
    preview.style.backgroundColor = args.currentValue.hex;
    preview.textContent = args.currentValue.hex ? args.currentValue.hex : 'No color';
  }
</script>
<style>
  .wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
  }
  #preview {
    border: 1px solid;
    font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
    font-size: 14px;
    height: 40px;
    line-height: 40px;
  }
</style>
```

NOCOLOR.CS

```
public ActionResult Nocolor()
{
    return View();
}
```

Custom no color

CSHTML

```

<div class='wrap'>
    <ul id="target" tabindex="0">
        <li class="e-item e-palette-item">

@Html.EJS().ColorPicker("element").Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette).Inline(true).ModeSwitcher(false).ShowButtons(false).Columns(4).BeforeTileRender("tileRender").PresetColors(@ViewBag.customColors).Change("onChange").Render()

        </li>
        <li class="e-item" id="no-color" tabindex="-1">
            <span class="e-menu-icon e-nocolor"></span>
            No color
        </li>
    </ul>
</div>
<div id="preview"></div>
<h4>Select color</h4>
@Html.EJS().SplitButton("splitbtn").IconCss("e-cp-icons e-picker-icon").Target("#target").Render()
</div>
<script>
    var preview;
    document.addEventListener("DOMContentLoaded", function () {
        preview = document.getElementById("preview");
        // Initially set the default color and value to preview.
        preview.style.backgroundColor = "#f44336";
        preview.textContent = "#f44336";
        // Binding click event for no color li.
        document.getElementById('no-color').onclick = function () {
            //sets color picker value property to null.
            ej.base.getInstance(document.getElementById("element"),
            ejs.inputs.ColorPicker).setProperties({ 'value': "" }, true);
            document.querySelector('.e-split-btn .e-picker-icon').style.borderBottomColor = "transparent";
            preview.textContent = "No color"
            preview.style.backgroundColor = "transparent";
        }
    });
    // Triggers while selecting colors from palette.
    function onChange(args) {
        var splitBtnInst =
        ej.base.getInstance(document.getElementById('splitbtn'),
        ejs.splitbuttons.SplitButton);
        document.querySelector(".e-split-btn .e-picker-icon").style.borderBottomColor = args.currentValue.hex;
        preview.style.backgroundColor = args.currentValue.hex;
        preview.textContent = args.currentValue.hex;
        if (splitBtnInst.element.getAttribute("aria-expanded")) {
            splitBtnInst.toggle();
            splitBtnInst.element.focus();
        }
    }
    // Triggers before rendering each palette tile.

```

```

function tileRender(args) {
    args.element.classList.add("e-custom-tile");
}
</script>
<style>
    /* Preview area styles */
    #preview {
        border: 1px solid;
        font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
        font-size: 14px;
        height: 40px;
        line-height: 40px;
        width: 100%;
    }
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
    @@font-face {
        font-family: 'paint';
        src:
            url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRIAAAEoAAAAVmNtYXNlODVAAABiAAAADZnbHlmiZD
+uwAAAcgAAADMaGVhZBKhHQAADQAAANmhoZWEHjANrAAAArAAAACRobXR4B+j/8wAAAYAAAAA
IbG9jYQBmAAAAAHAAAAABmlheHABDgBKAAABCAAAACBuYW1ln6hzswAAApQAAAINcG9zdEkLMmU
AAASkAAAAANgABAAADUv9qAFoEAP/z//4D6gABAAAAAAAAAAAAAAAAAAgABAAAAAQAAZfc6F8
PPPUACwPoAAAAANfSn9kAAAAA19Kf2f/z//wD6gPhAAAACAACAAAAAAAAAAEAAAACAD4AAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPhAJYAAAABAAAAAAAAABAAAAAPo//M
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAEAAAA
AAAAAZgAAAAAL/8//8A+oD4QAKAD0AAAEBGgeATclJiQHJTMmNjceARcVJx4BBx4BFQ4BIiYnNDY
3PgEvAS4BIw4BBwEGHgI3AT4BLwE1LgEnDgEDeIRlCgulCxP+8RT+GyYDQFxoZQwTBQEDDxEBJzo
nAREOCQkPJQ4cDBcdAf6oG1a3nx8BWQ4RHKADeG1oWwHTLHVwYVml6Kx1BHEqfwYFqWUHEx4tDAo
cEx0nJx0RHgoVUDQpDgsBFAH+px2guFUaAvkNoiCgCXnhCAWOAAAAAAAAAEgDeAAEAAAAAAAAQA
AAEAAAAAAAAEABQABAAEAAAAAAAAIABwAGAAEAAAAAAAAAMABQANAAEAAAAAAAAQABQASAAEAAAAAAU
ACwAXAAEAAAAAAAAAYABQAIAAEAAAAAAAAoALAAEAAAAAAAAAsAEgBTAAMAAQQJAAAAAgBlaAMAAQQ
JAAEACgBnAAMAAQQJAAIADgBxAAMAAQQJJAAMACgB/AAMAAQQJJAACgCJAAMAAQQJJAUAfGCTAAM
AAQQJJAAYACgCpAAMAAQQJJAoAWACzAAMAAQQJJAAsAJAELIHBhaW50UmVndWxhcncBhaW50cGFpbmR
WZXJzaW9uIDEuMHBhaW50Rm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Zlc2l2b20AIABwAGEAaQB
uAHQAUGBlAGcAdQBsaGEAcgBwAGEAaQBwAHQA
AcABhAGkAbgB0AFYAZQByAHMAaQBvAG4AIAAxAAC4AMABwAGEAaQBwAHQAQAgBvAG4AdAAgAGcAZQB
uAGUAACgBhAHQAQZQBkACAAQZBzAGkAbgBnACAAUwB5AG4AYwBmAHUAACwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMADABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAACwBpAG8AbgAuAGMabwBtAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMADHBhaW50LWJlY2tldAAAAA=)
format('truetype');
        font-weight: normal;
        font-style: normal;
    }
    .e-cp-icons {
        font-family: 'paint' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
    }

```


</style>

CUSTOM-NOCOLOR.CS

$$\{$$

```

List<object> custom = new List<object>();
custom.Add(new
{
    Custom1 = new string[] { "#f44336", "#e91e63", "#9c27b0",
"#673ab7", "#2196f3", "#03a9f4", "#00bcd4",
"#009688", "#8bc34a", "#cddc39", "#ffeb3b", "#ffc107" }
});
ViewBag.customColors = custom[0];
return View();
}

```

Note: [View Sample in GitHub.](#)

Disabled in Color Picker Control

To achieve disabled state in ColorPicker, set the [disabled](#) property to `true`. The ColorPicker pop-up cannot be accessed in disabled state.

CSHTML

```

<div class='wrap'>
    <h4>Disabled state</h4>
    @Html.EJS().ColorPicker("element").Disabled(true).Render()
</div>
<style>
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }
</style>

```

DISABLED.CS

```

public ActionResult Disabled()
{
    return View();
}

```

Note: [View Sample in GitHub.](#)

Migration from Essential JS 1

This article describes the API migration process of ColorPicker component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **property:** `value`

 `@Html.EJ().ColorPicker("colorpicker").Value("#278787")` | **property:** `value`

 `@Html.EJS().ColorPicker("colorpicker").Value("#278787").Render()` |

| Inline mode color picker | **property:** *displayInline*

@Html.EJ().ColorPicker("colorpicker").DisplayInline(true).TagName("div") | **property:** *inline*

 @Html.EJS().ColorPicker("colorpicker").Inline(true).Render() |

| Adding custom class | **property:** *cssClass*

@Html.EJ().ColorPicker("colorpicker").CssClass("e-custom") | **property:** *cssClass*

@Html.EJS().ColorPicker("colorpicker").CssClass("e-custom").Render() |

| Disable the ColorPicker component | **property:** *enabled*

@Html.EJ().ColorPicker("colorpicker").Enabled(false) | **property:** *disabled*

@Html.EJS().ColorPicker("colorpicker").Disabled(true).Render() |

| To display custom text in button elements | **property:** *buttonText*

@Html.EJ().ColorPicker("colorpicker").ColorPickerButtonText(color=>

color.Apply("Apply").Cancel("Cancel")) | Not Applicable |

| To display customized text or content when mouse over the color picker elements | **property:**
tooltipText

 @Html.EJ().ColorPicker("colorpicker").ColorPickerTooltipText(tooltip=>

 tooltip.Switcher("Switch").Sandy("arenoso")) | Not Applicable |

| Disable / hide opacity | **property:** *enableOpacity*

@Html.EJ().ColorPicker("colorpicker").EnableOpacity(false) | **property:** *enableOpacity*

@Html.EJS().ColorPicker("colorpicker").enableOpacity(false).Render() |

| ColorPicker Button mode | **property:** *buttonMode*

@Html.EJ().ColorPicker("colorpicker").ButtonMode(ButtonMode.Dropdown) | Not Applicable |

| To show / hide the control (apply / cancel) buttons | **property:** *showApplyCancel*

@Html.EJ().ColorPicker("colorpicker").ShowApplyCancel(false) | **property:** *showButtons*

 @Html.EJS().ColorPicker("colorpicker").ShowButtons(false).Render() |

| To show / hide the clear button | **property:** *showClearButton*

@Html.EJ().ColorPicker("colorpicker").ShowClearButton(false) | Not Applicable |

| Show / hide the mode (picker / palette) switcher | **property:** *showSwitcher*

@Html.EJ().ColorPicker("colorpicker").ShowSwitcher(false) | **property:** *modeSwitcher*

@Html.EJS().ColorPicker("colorpicker").ModeSwitcher(false).Render() |

| To show / hide the preview area | **property:** *showPreview*

@Html.EJ().ColorPicker("colorpicker").ShowPreview(false) | Not Applicable |

| To show / hide the recent selected color list | **property:** *showRecentColors*

@Html.EJ().ColorPicker("colorpicker").ShowRecentColors(true) | Not Applicable |

| To show / hide the color picker slider tooltip | **property:** *showTooltip*

@Html.EJ().ColorPicker("colorpicker").ShowTooltip(false) | Not Applicable |

| Custom icon in dropdown control color area | **property:** *toolIcon*

@Html.EJ().ColorPicker("colorpicker").ToolIcon("e-font-icon") | Not Applicable |

| ColorPicker mode | **property:** *modelType*

@Html.EJ().ColorPicker("colorpicker").(ModelType.Picker) | **property:** *mode*

@Html.EJS().ColorPicker("colorpicker")

 .Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette)).Render() |
 | Opacity value | **property:** *opacityValue*

 @Html.EJ().ColorPicker("colorpicker").OpacityValue(80) | Not Applicable |
 | Number of columns in color palette | **property:** *columns*

 @Html.EJ().ColorPicker("colorpicker").Columns(10) | **property:** *columns*

 @Html.EJS().ColorPicker("colorpicker").Columns(15).Render() |
 | Custom colors | **property:** *palette*

 @Html.EJ().ColorPicker("colorpicker").Palette(PaletteType.CustomPalette)

 .ModelType(ModelType.Palette).Custom(@ViewBag.colors)
 List<String> colors = new
 List<string> { "ffffff", "ffccff", "ff99ff", "ff66ff", "ff33ff", "ff00ff", "ccffff", "ccccff", "cc99ff", "cc66ff",
 "cc33ff", "cc00ff", "99ffff", "99ccff", "9999ff", "9966ff", "9933ff", "9900ff", "ffffcc", "ffcccc" };

 ViewBag.colors = colors; | **property:** *presetColors*

 @Html.EJS().ColorPicker("colorpicker").PresetColors(@ViewBag.colors)
 .Render()

 List<object> custom = new List<object>() {
 Custom = new string[] { "ffffff", "ffccff", "ff99ff",
 "ff66ff", "ff33ff", "ff00ff", "ccffff", "ccccff", "cc99ff", "cc66ff", "cc33ff", "cc00ff", "99ffff", "99ccff",
 "9999ff", "9966ff", "9933ff", "9900ff", "ffffcc", "ffcccc" }
 };
 ViewBag.colors = colors[0]; |
 | Rendering palette from the predefined set of palettes | **property:** *presetType*

 @Html.EJ().ColorPicker("colorpicker").ModelType(ModelType.Palette)

 .PresetType(PresetsType.FlatColors) | Not Applicable |
 | No color option in color palette | Not Applicable | **property:** *noColor*

 @Html.EJS().ColorPicker("colorpicker").NoColor(true).ModeSwitcher(false)

 .Mode(Syncfusion.EJ2.Inputs.ColorPickerMode.Palette)).Render() |
 | Localization | **property:** *locale*

 @Html.EJ().ColorPicker("colorpicker").Locale("zh-CN")

 scripts:
 ej.ColorPicker.Locale["zh-CN"] = {
 buttonText: {

 apply: "应用",
 cancel: "取消",
 swatches: "色板"

 },
 tooltipText: {
 switcher: "切换器",

 addButton: "添加颜色",
 basic: "基本"
 }
 } | **property:** *locale*

 @Html.EJS().ColorPicker("colorpicker").Locale("ar").Render()
 scripts:

 ej.base.L10n.load({
 'ar': {
 "colorpicker": {

 "Apply": "تطبيق",
 "Cancel": "إلغاء",

 "ModeSwitcher": "مفتاح كهربائي الوضع"
 }
 }
 }); |
 | Right to left | **property:** *enableRTL*

 @Html.EJ().ColorPicker("colorpicker").EnableRTL(true) | **property:** *enableRtl*

 @Html.EJS().ColorPicker("colorpicker").EnableRtl(true).Render() |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

— — — — —

| Method to open color picker popup | **Method:** `show`
 `@Html.EJ().ColorPicker("colorpicker")`
 `var colorPickerObj =`

`$("#colorpicker").data("ejColorPicker");
 colorPickerObj.show(); | Method: toggle

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];
 colorPickerObj.toggle(); |`

| Method to close color picker popup | **Method:** *hide*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.hide(); | **Method:** *toggle*

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];
 colorPickerObj.toggle(); |

| Enable the color picker control | **Method:** *enable*

 @Html.EJ().ColorPicker("colorpicker")

 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.enable(); | Not
 Applicable |

| Disables the color picker control | **Method:** *disable*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.disable(); | Not Applicable |

| Method returns the selected color value as hex code | **Method:** *getValue*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.getValue(); | **Method:** *getValue*

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];
 colorPickerObj.getValue(); |

| Method returns the selected color value in RGB format | **Method:** *getColor*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.getColor(); | **Method:** *getValue*

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];
 colorPickerObj.getValue(null, 'RGB'); |

| Method convert the color value from hexCode to RGB | **Method:** *hexCodeToRGB*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.hexCodeToRGB("#278787"); | **Method:**
getValue

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];
 colorPickerObj.getValue("#278787",
 'RGB'); |

| Method convert the color value from RGB to Hex code | **Method:** *RGBToHEX*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.RGBToHEX({r:38,g:133,b:133}); | **Method:**
getValue

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];

 colorPickerObj.getValue("rgb(38,133,133)", 'Hex'); |

| Method convert the color value from RGB to HSV | **Method:** *RGBToHSV*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.RGBToHSV({h:230,s:98,v:98}); | **Method:**
getValue

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];

 colorPickerObj.getValue("rgb(180,71.1,52.9)", 'HSV'); |

| Method convert the color value from HSV to RGB | **Method:** *HSVToRGB*

 @Html.EJ().ColorPicker("colorpicker")
 var colorPickerObj =
 \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.HSVToRGB({h:230,s:98,v:98}); | **Method:**
getValue

 @Html.EJS().ColorPicker("colorpicker").Render()
 var colorPickerObj =
 document.getElementById("colorpicker").ej2_instances[0];

colorPickerObj.getValue("hsv(180,71.1,52.9)", 'RGB'); |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Event triggers before opening the ColorPicker popup | Not Applicable | **Event:** *beforeOpen*

 @Html.EJS().ColorPicker("colorpicker").BeforeOpen("beforeOpen").Render()
 function
 beforeOpen(args) {
 / code block */
 } |

| Event triggers before closing the ColorPicker popup | Not Applicable | **Event:** *beforeClose*

 @Html.EJS().ColorPicker("colorpicker").BeforeClose("beforeClose").Render()
 function
 beforeClose(args) {
 / code block */
 } |

| Event triggers after opening the ColorPicker popup | **Event:** *open*

 @Html.EJ().ColorPicker("colorpicker").Open("open")
 function open(args) {

 / code block /
 } | **Event:** *open*

 @Html.EJS().ColorPicker("colorpicker").Open("open").Render()
 function open(args) {

 / code block /
 } |

| Event triggers after closing the ColorPicker popup | **Event:** *close*

 @Html.EJ().ColorPicker("colorpicker").Close("close")
 function close(args) {

 / code block */
 } | Not Applicable |

| Event triggers once the component rendering is completed | **Event:** *create*

 @Html.EJ().ColorPicker("colorpicker").Create("create")
 function create(args) {

 / code block /
 } | **Event:** *created*

 @Html.EJS().ColorPicker("colorpicker").Created("created").Render()
 function created() {

 / code block /
 } |

| Event triggers once the color picker control is destroyed | **Event:** *destroy*

 @Html.EJ().ColorPicker("colorpicker").Destroy("destroy")
 function destroy(args) {

 / code block */
 } | Not Applicable |

| Event triggers before Switching between Picker / Palette mode | Not Applicable | **Event:**
beforeModeSwitch

 @Html.EJS().ColorPicker("colorpicker").BeforeModeSwitch("beforeModeSwitch").Render()

 function beforeModeSwitch(args) {
 / code block */
 } |

| Event triggers after color value has been selected | **Event:** *select*

 @Html.EJ().ColorPicker("colorpicker").Select("select")
 function select(args) {

 / code block /
 } | **Event:** *select*

 @Html.EJS().ColorPicker("colorpicker").Select("select").Render()
 function select(args) {

 / code block /
 } |

| Event triggers after color value has been changed | **Event:** *change*

 @Html.EJ().ColorPicker("colorpicker").Change("change")
 function change(args) {


```
&#160; &#160; / code block / <br/> } | Event: change <br/><br/>
@Html.EJS().ColorPicker("colorpicker").Change("change").Render() <br/> function change(args) {
<br/> &#160; &#160; / code block / <br/> } |
```

ComboBox

Getting Started with ASP.NET MVC ComboBox Control

This section briefly explains about how to include [ASP.NET MVC ComboBox](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
,
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
,
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
```

```

...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>

```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```

<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>

```

Add ASP.NET MVC ComboBox control

Now, add the Syncfusion ASP.NET MVC ComboBox control in `~/Views/Home/Index.cshtml` page.

CSHTML

```

@using Syncfusion.EJ2
@model List<string>
@Html.EJS().ComboBox("games").Placeholder("Select a
game").DataSource((IEnumerable<object>)Model).Render()

```

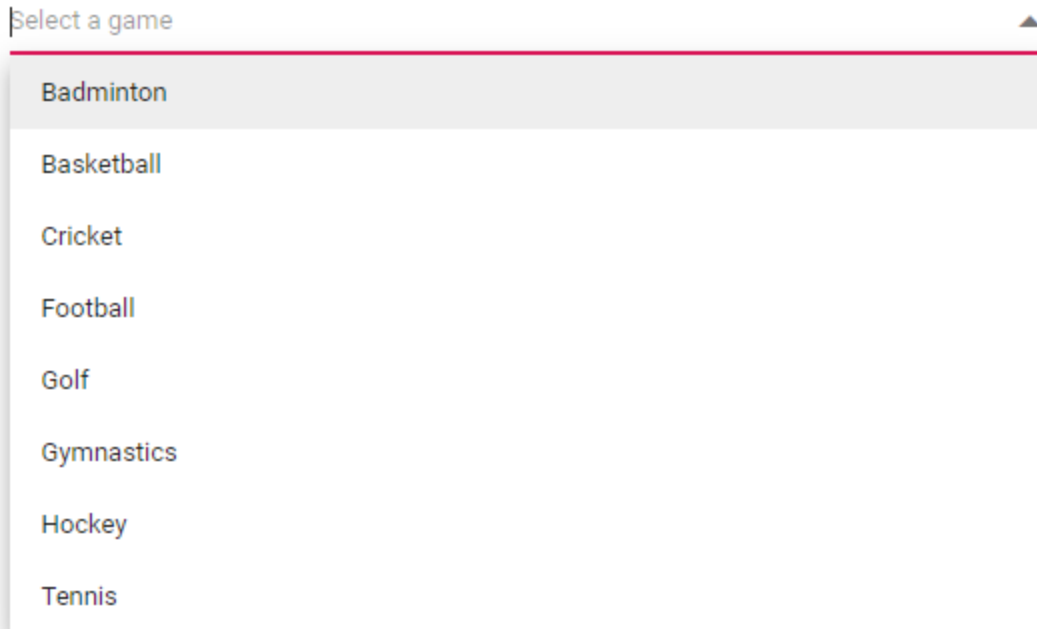
HOMECONTROLLER.CS

```

public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}

```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC ComboBox control will be rendered in the default web browser.



Custom values

The ComboBox allows the user to give input as custom value which is not required to present in predefined set of values. By default, this support is enabled by [allowCustom](#) property. In this case, both text field and value field considered as same. The custom value will be sent to post back handler when a form is about to be submitted.

CSHTML

```
@Html.EJS().ComboBox("games").AllowCustom(true).Placeholder("Select a game").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Configure the popup list

By default, the width of the popup list automatically adjusts according to the ComboBox input element's width, and the height of the popup list has '300px'.

The height and width of the popup list can also be customized using the [popupHeight](#) and [popupWidth](#) properties respectively.

In the following sample, popup list's width and height are configured.

CSHTML

```
@Html.EJS().ComboBox("games").AllowCustom(true).Placeholder("Select a
game").PopupWidth("300px").PopupHeight("230px").DataSource((IEnumerable<obje
ct>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Note: [View Sample in GitHub.](#)

See also

- [How to bind the data](#)

Data Binding in Combo Box Control

The ComboBox loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of [array](#) or [DataManager](#).

The ComboBox also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of [DataManager](#) adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

Note: When binding complex data to the ComboBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of simple data

The ComboBox has supported to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

CSHTML

```
@Html.EJS().ComboBox("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Re
nder()
```

ARRAYOFSTRINGS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult arrayofstring()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball",
"Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}
```

2. Array of JSON data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Vegetable** column from complex data has been mapped to the **value** field.

CSHTML

```
@Html.EJS().ComboBox("Vegetables").Placeholder("Select a
Vegetables").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.da
ta).Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value =
"Vegetable" }).Render()
```

ARRAYOFOBJECTS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult arrayofobjects()
        {
            ViewBag.data = new Vegetables().VegetablesList();
            return View();
        }
    }
}
```

3. Array of Complex data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Code.Id` column and `Country.CountryId` column from complex data have been mapped to the `value` field and `text` field, respectively.

CSHTML

```
@Html.EJS().ComboBox("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<Object>)ViewBag.data).Fi
elds(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text =
"Country.CountryId", Value = "Code.Id" }).Render()
```

COMPLEXDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxControleler : Controller
    {
        public ActionResult complexdata()
        {
            ViewBag.data = new Complex().GetData();
            return View();
        }
    }
}
```

Binding remote data

The ComboBox supports retrieval of data from remote data services with the help of [DataManager](#) component. The [Query](#) property is used to fetch data from the database and bind it to the ComboBox.

In the following sample, displayed first 6 contacts from the `customer` table of `Northwind` Data Service.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{
    Text = "ContactName",
    Value = "CustomerID"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
```

REMOTEDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult remotedata()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Bind to URL Adaptor

The ComboBox supports retrieval of data from URL adaptor.

CSHTML

```
@Html.EJS().ComboBox("games").Placeholder("Select a
Country").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("/ComboBox/UrlDatasource").Adaptor("UrlAdaptor").CrossDomain(
true)).Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{
    Value = "ShipCountry"
}).Render()
```

URLDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult UrlDatasource([FromBody]Data dm)
        {
            var val = OrdersDetails.GetAllRecords();
            var Data = val.ToList();
            var count = val.Count();
        }
    }
}
```

```

        if (dm.where != null)
        {
            Data = (from cust in Data
                    where
cust.ShipCountry.ToLower().StartsWith(dm.@where[0].value.ToString())
                    select cust).ToList();
        }
        if (dm.take != 0)
            Data = Data.Take(dm.take).ToList();
        return Json(Data);
    }
    public class Data
    {
        public int take { get; set; }
        public List<Wheres> where { get; set; }
    }
    public class Wheres
    {
        public string field { get; set; }
        public bool ignoreAccent { get; set; }
        public bool ignoreCase { get; set; }
        public bool isComplex { get; set; }
        public string value { get; set; }
        public string Operator { get; set; }
    }
    public class OrdersDetails
    {
        public static List<OrdersDetails> order = new
List<OrdersDetails>();
        public OrdersDetails()
        {
        }
        public OrdersDetails(int OrderID, string CustomerId, int
EmployeeId, double Freight, bool Verified, DateTime OrderDate, string
ShipCity, string ShipName, string ShipCountry, DateTime ShippedDate, string
ShipAddress)
        {
            this.OrderID = OrderID;
            this.CustomerID = CustomerId;
            this.EmployeeID = EmployeeId;
            this.Freight = Freight;
            this.ShipCity = ShipCity;
            this.Verified = Verified;
            this.OrderDate = OrderDate;
            this.ShipName = ShipName;
            this.ShipCountry = ShipCountry;
            this.ShippedDate = ShippedDate;
            this.ShipAddress = ShipAddress;
        }
        public static List<OrdersDetails> GetAllRecords()
        {
            if (order.Count() == 0)
            {
                int code = 10000;
                for (int i = 1; i < 10; i++)
                {

```

```

        order.Add(new OrdersDetails(code + 1, "ALFKI", i +
0, 2.3 * i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro",
"Denmark", new DateTime(1996, 7, 16), "Kirchgasse 6"));
        order.Add(new OrdersDetails(code + 2, "ANATR", i +
2, 3.3 * i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha",
"Brazil", new DateTime(1996, 9, 11), "Avda. Azteca 123"));
        order.Add(new OrdersDetails(code + 3, "ANTON", i +
1, 4.3 * i, true, new DateTime(1957, 11, 30), "Cholchester",
"Frankenversand", "Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave.
Bolívar #65-98 Llano Largo"));
        order.Add(new OrdersDetails(code + 4, "BLONP", i +
3, 5.3 * i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel",
"Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
        order.Add(new OrdersDetails(code + 5, "BOLID", i +
4, 6.3 * i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes",
"Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
        code += 5;
    }
}
return order;
}
}

public int? OrderID { get; set; }
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
public bool Verified { get; set; }
public DateTime OrderDate { get; set; }
public string ShipName { get; set; }
public string ShipCountry { get; set; }
public DateTime ShippedDate { get; set; }
public string ShipAddress { get; set; }
}
}
}

```

Web API Adaptor

Use the **WebApiAdaptor** to bind ComboBox with Web API created using OData.

CSHTML

```

<div id='web-data' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().ComboBox("games").DataSource(dataManger =>
dataManger.Url("/api/ComboBox/").Adaptor("WebApiAdaptor").Render()
    </div>
</div>

```

WEBAPI.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;

```

```

using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    [Route("api/[controller]")]
    public class ComboBoxController : Controller
    {
        List<string> game = new List<string>();
        [HttpGet]
        public List<string> Get()
        {
            game.Add("Badminton");
            game.Add("Basketball");
            game.Add("Cricket");
            game.Add("Golf");
            game.Add("Gymnastics");
            game.Add("Tennis");
            game.Add("Hockey");
            return game;
        }
    }
}

```

Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager.

CSHTML

```

<div id='o-data' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().ComboBox("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders/").Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
    {
        Value = "CustomerID"
    }).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
    </div>
</div>

```

ODATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {

```



```

        public ActionResult odata()
        {
            return View();
        }
    }
}

```

Offline mode

To avoid post back for every action, set the ComboBox to load all data on initialization and make the actions process in client-side. To enable this behavior, use the **Offline** property of [DataManager](#).

CSHTML

```

<div id='o-odata' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().ComboBox("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("http://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders
/").Offline(true).Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
    {
        Value = "CustomerID"
    }).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
    </div>
</div>

```

OFFLINE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public IActionResult offline()
        {
            return View();
        }
    }
}

```

See also

- [How to acheive cascading](#)
- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Templates in Combo Box Control

The ComboBox has been provided with several options to customize each list item, group title, selected value, header, and footer elements. It uses the Essential JS 2 Template engine to compile and render the elements properly.

Item template

The content of each list item within the ComboBox can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(obj => obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("<span><span class='name'>${FirstName}</span><span class='city'>${City}</span></span>").Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "FirstName" }).Query("new ej.data.Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6)").Render()<style> .city { right: 15px; position: absolute; } </style>
```

ITEMTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult Itemtemplate()
        {
            return View();
        }
    }
}
```

Group template

The group header title under which appropriate sub-items are categorized can also be customized with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).GroupTemplate("<strong>${City}</strong>").Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{
    Value = "FirstName",
    GroupBy = "City"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6).where(new ej.data.Predicate('City', 'equal', 'london').or('City', 'equal', 'seattle'))").Render()
```

GROUPTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult grouptemplate()
        {
            return View();
        }
    }
}
```

Header template

The header element is shown statically at the top of the popup list items within the ComboBox, and any custom element can be placed as a header element using the [headerTemplate](#) property.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("<span class='item'><span class='name'>${FirstName}</span><span class='city'>${City}</span></span>").HeaderTemplate("<span class='head'><span class='name'>Name</span><span
```

```
class='city'>City</span></span>").Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{
    Value = "FirstName",
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
    .head, .item {
        display: table;
        width: 100%;
        margin: auto;
    }
    .head {
        height: 40px;
        font-size: 15px;
        font-weight: 600;
    }
    .name, .city {
        display: table-cell;
        vertical-align: middle;
        width: 50%;
    }
    .head .name {
        text-indent: 16px;
    }
    .head .city {
        text-indent: 10px;
    }
</style>
```

HEADERTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult headertemplate()
        {
            return View();
        }
    }
}
```

Footer template

The ComboBox has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

CSHTML

```
@Html.EJS().ComboBox("games").DataSource((IEnumerable<object>)ViewBag.data).
Placeholder("Select a game").PopupHeight("270px").FooterTemplate("<span
class='foot'> Total list items: " + 5 + "</span>").Render()
<style>
    .foot {
        text-indent: 1.2em;
        display: block;
        font-size: 15px;
        line-height: 40px;
        border-top: 1px solid #e0e0e0;
    }
</style>
```

FOOTERTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public IActionResult footertemplate()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball",
"Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}
```

No records template

The ComboBox is provided with support to custom design the popup list content when no data is found and no matches are found on search with the help of [noRecordsTemplate](#) property.

CSHTML

```
@Html.EJS().ComboBox("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).No
RecordsTemplate("<span class='norecord'> NO DATA AVAILABLE</span>").Render()
```

NORECORDSTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
```

```

{
    public ActionResult norecords()
    {
        ViewBag.data = new string[] { };
        return View();
    }
}

```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

CSHTML

```

@Html.EJS().ComboBox("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svcs/").Ad
aptor("ODataV4Adaptor").CrossDomain(true)).ActionFailureTemplate("<span
class='action-failure'> Data fetch get fails</span>").Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{
    Value = "FirstName"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName']).take(6)").Render()

```

ACTIONFAILURETEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult actionfailure()
        {
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

See also

- [How to acheive filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)

Virtualization in ComboBox Component

ComboBox virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a ComboBox activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the Combobox component. When a key is typed, the focus is moved to the respective element in the open popup state. In the closed popup state, the popup opens, and focus is moved to the respective element in the popup list based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

When the enableVirtualization property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The Combobox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `id` column and `text` column from complex data have been mapped to the `value` field and `text` field, respectively.

CSHTML

```
@Html.EJS().ComboBox("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>) ViewBag.data).En
ableVirtualization(true).AllowFiltering(false).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "ID", Text="Text"
}).Render()
```

VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
```

```

{
    public ActionResult virtualscroll()
    {
        ViewBag.data = new Record().RecordModelList();
        return View();
    }
}

```

Binding remote data

The Combobox supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

CSHTML

```

@Html.EJS().ComboBox("records").Placeholder("Select a
item").PopupHeight("200px").DataSource(obj
=>obj.Url("https://ej2services.syncfusion.com/aspnet/development/api/orders"
).CrossDomain(true).Adaptor("WebApiAdaptor")).EnableVirtualization(true).All
owFiltering(true).Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{ Text = "OrderID", Value = "OrderID" }).Render()

```

VIRTUALSCROLL.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult virtualscroll()
        {
            ViewBag.data = new Record().RecordModelList();
            return View();
        }
    }
}

```

Grouping

The Combobox component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the **groupBy** field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to

retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding virtualization, enhancing performance and responsiveness.

The following sample shows the example for Grouping with Virtualization.

CHTML

```
@Html.EJS().ComboBox("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).En
ableVirtualization(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { GroupBy = "Group", Value =
"ID", Text="Text" }).Render()
```

VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult virtualscroll()
        {
            ViewBag.data = new Record().RecordModelList();
            return View();
        }
    }
}
```

Filtering with Virtualization

The ComboBox component supports Filtering with Virtualization. The ComboBox includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the DropDownList sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

CHTML

```
@Html.EJS().ComboBox("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).En
ableVirtualization(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "ID", Text="Text"
}).Render()
```

VIRTUALSCROLL.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult virtualscroll()
        {
            ViewBag.data = new Record().RecordModelList();
            return View();
        }
    }
}

```

Grouping in Combo Box Control

The ComboBox supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

CSHTML

```

@Html.EJS().ComboBox("Vegetable").Placeholder("Select a
Vegetable").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data)
.Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value =
"Vegetable", GroupBy = "Category" }).Render()

```

VEGETABLES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Vegetables
    {
        public string Vegetable { get; set; }
        public string Category { get; set; }
        public string Id { get; set; }
        public List<Vegetables> VegetablesList()
        {
            List<Vegetables> veg = new List<Vegetables>();
            veg.Add(new Vegetables { Vegetable = "Cabbage", Category =
"Leafy and Salad", Id = "item1" });
            veg.Add(new Vegetables { Vegetable = "Chickpea", Category =
"Beans", Id = "item2" });
            veg.Add(new Vegetables { Vegetable = "Garlic", Category = "Bulb
and Stem", Id = "item3" });
        }
    }
}

```

```
veg.Add(new Vegetables { Vegetable = "Green bean", Category =
"Beans", Id = "item4" });
veg.Add(new Vegetables { Vegetable = "Horse gram", Category =
"Beans", Id = "item5" });
veg.Add(new Vegetables { Vegetable = "Nopal", Category = "Bulb
and Stem", Id = "item6" });
veg.Add(new Vegetables { Vegetable = "Onion", Category = "Bulb
and Stem", Id = "item7" });
veg.Add(new Vegetables { Vegetable = "Pumpkins", Category =
"Leafy and Salad", Id = "item8" });
veg.Add(new Vegetables { Vegetable = "Spinach", Category =
"Leafy and Salad", Id = "item9" });
veg.Add(new Vegetables { Vegetable = "Wheat grass", Category =
"Leafy and Salad", Id = "item10" });
veg.Add(new Vegetables { Vegetable = "Yarrow", Category = "Leafy
and Salad", Id = "item11" });
return veg;
}
}
```

Note: [View Sample in GitHub.](#)

Customization

The grouping header is also provided with customization option. This allows custom designing using the `groupTemplate` property for both inline and fixed headers as referred here:

[Group Template support to ComboBox.](#)

Filtering in Combo Box Control

The ComboBox has built-in support to filter data items when `allowFiltering` is enabled. The filter operation starts as soon as you start typing characters in the control.

To display filtered items in the popup, filter the required data and return it to the ComboBox via `updateData` method by using the `filtering` event.

The following sample illustrates how to query the data source and pass the data to the ComboBox through the `updateData` method in `filtering` event.

CSHTML

```
@Html.EJS().ComboBox("country").Placeholder("Select a COUNTRY").PopupHeight(
"230px").DataSource((IEnumerable<object>)ViewBag.data).AllowFiltering(true).
Filtering("onfiltering").Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "Name" }).Render()
<script>
    function onfiltering(e) {
        var query = new ej.data.Query();
        query = (e.text !== '') ? query.where('Name', 'startswith',
e.text, true) : query;

e.updateData(@Html.Raw(JsonConvert.SerializeObject(ViewBag.data)), query)
    }
</script>
```

COUNTRIES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Countries
    {
        public string Name { get; set; }
        public string Code { get; set; }
        public List<Countries> CountriesList()
        {
            List<Countries> country = new List<Countries>();
            country.Add(new Countries { Name = "Australia", Code = "AU" });
            country.Add(new Countries { Name = "Bermuda", Code = "BM" });
            country.Add(new Countries { Name = "Canada", Code = "CA" });
            country.Add(new Countries { Name = "Cameroon", Code = "CM" });
            country.Add(new Countries { Name = "Denmark", Code = "DK" });
            country.Add(new Countries { Name = "France", Code = "FR" });
            country.Add(new Countries { Name = "Finland", Code = "FI" });
            country.Add(new Countries { Name = "Germany", Code = "DE" });
            country.Add(new Countries { Name = "Greenland", Code = "GL" });
            country.Add(new Countries { Name = "Hong Kong", Code = "HK" });
            country.Add(new Countries { Name = "India", Code = "IN" });
            country.Add(new Countries { Name = "Italy", Code = "IT" });
            country.Add(new Countries { Name = "Japan", Code = "JP" });
            country.Add(new Countries { Name = "Mexico", Code = "MX" });
            country.Add(new Countries { Name = "Norway", Code = "NO" });
            country.Add(new Countries { Name = "Poland", Code = "PL" });
            country.Add(new Countries { Name = "Switzerland", Code = "CH"
});
            country.Add(new Countries { Name = "United Kingdom", Code = "GB"
});
            country.Add(new Countries { Name = "United States", Code = "US"
});
            return country;
        }
    }
}

```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the ComboBox. This can be done by manual validation within the filter event handler.

CSHTML

```

@Html.EJS().ComboBox("customers").Placeholder("Select a
customer").Filtering("onfiltering").AllowFiltering(true).PopupHeight("200px"
).DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
    "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text = "ContactName", Value
= "CustomerID" }).Query("new

```

```
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)".Render()
<script>
    function onfiltering(e) {
        var CBObj = document.getElementById("customers").ej2_instances[0];
        // load overall data when search key empty.
        if (e.text == '' && e.text.length < 3) {
            e.updateData(CBObj.dataSource);
        }
        let query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
        query = (e.text != '' && e.text.length >= 3) ?
query.where('ContactName', 'startswith', e.text, true) : query;
        e.updateData(CBObj.dataSource, query);
    }
</script>
```

FILTERLIMIT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult filteringlimit()
        {
            return View();
        }
    }
}
```

Change the filter type

While filtering, you can change the filter type to **contains**, **startsWith**, or **endsWith** for string type within the filter event handler.

In the following examples, data filtering is done with **endsWith** type.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a
customer").AllowFiltering(true).Filtering("onfiltering").PopupHeight("200px"
).DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
    "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text = "ContactName", Value
= "CustomerID" }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)".Render()
<script>
```

```
function onfiltering(e) {
    var CBObj = document.getElementById("customers").ej2_instances[0];
    // load overall data when search key empty.
    if (e.text == '')
        e.updateData(CBObj.dataSource);
    else {
        var query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
        query = (e.text != '') ? query.where('ContactName', 'endswith',
e.text, true) : query;
        e.updateData(CBObj.dataSource, query);
    }
}
</script>
```

FILTERTYPE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult filtertype()
        {
            return View();
        }
    }
}
```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the **where** clause.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a
customer").Filtering("onfiltering").AllowFiltering(true).PopupHeight("200px"
).DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
    "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text = "ContactName", Value
= "CustomerID" }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
function onfiltering(e) {
    var CBObj = document.getElementById("customers").ej2_instances[0];
    // load overall data when search key empty.
    if (e.text == '')
```

```

        e.updateData(CBObj.dataSource);
    else {
        var query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
        query = (e.text != '') ? query.where('ContactName',
'startswith', e.text, false) : query;
        e.updateData(CBObj.dataSource, query);
    }
}
</script>

```

CASESENSITIVE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult casesensitive()
        {
            return View();
        }
    }
}

```

Diacritics filtering

ComboBox supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

CSHTML

```

@Html.EJS().ComboBox("list").DataSource((string[])ViewBag.data).Placeholder(
"e.g: aero").IgnoreAccent(true).AllowFiltering(true).Render()

```

DIACRITICS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult diacritics()
        {

```

```
ViewBag.data = new string[] { "Aeróbics", "Aeróbics en Agua",
"Aerografía", "Aerodelaje", "Águilas", "Ajedrez", "Ala Delta", "Álbumes de
Música", "Alusivos", "Análisis de Escritura a Mano" };
return View();
}
}
```

Note: [View Sample in GitHub.](#)

See also

- [How to achieve autofill while filtering](#)
- [How to group the data using header](#)

Localization in Combo Box Control

The Localization library allows to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the ComboBox.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use `load` function of **L10n** class.

In the following sample, French culture is set to the ComboBox and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

CSHTML

```
@Html.EJS().ComboBox("customers").Placeholder("Select a
customer").PopupHeight("200px").Locale("fr-BE").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings
{
    Value = "ContactName"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(0)").Render()
<script>
    ej.base.L10n.load({
        'fr-BE': {
            'dropdowns': {
                'noRecordsTemplate': "Aucun enregistrement trouvé",
                'actionFailureTemplate': "Modèle d'échec d'action"
            }
        }
    });
```



```
</script>
```

LOCALIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult localization()
        {
            return View();
        }
    }
}
```

Note: [View Sample in GitHub.](#)

See also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}
`
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
```

```
.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}
、
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-
input-focus::after {
background: #c000ff;
}
、
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-
success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}
、
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css
.e-input-group.e-control-wrapper .e-input[disabled] {
-webkit-text-fill-color: #0d9133;
}
、
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css
```

```
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
```

```
background-color: #2319b8;
```

```
}
```

```
.e-ddl.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-float-
input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
```

```
color: #2319b8;
```

```
}
```

```
,
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css
```

```
.e-ddl.e-input-group input.e-input::placeholder {
```

```
color: red;
```

```
}
```

```
,
```

Customizing the placeholder to add mandatory indicator(*)

Use the following CSS to add the mandatory indicator * to the float label element.

```
`css
```

```
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after {
```

```
content: "*";
```

```
color: red;
```

```
}
```

```
,
```

Customizing the text selection color

Use the following CSS to customize the selection color of text and background.

```
`css
```

```
.e-ddl.e-input-group input.e-input::selection {
```

```
color: red;
```

```
background: yellow;
```

```
}
```

```
,
```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
`
```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
`
```

Accessibility in Combo Box Control

The ComboBox control has been designed, keeping in mind the **WAI-ARIA** specifications, and applies the WAI-ARIA roles, states, and properties along with **keyboard support**. This control is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The ComboBox component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ComboBox component is outlined below.

Accessibility Criteria	Compatibility
--	--
WCAG 2.2 Support	
Section 508 Support	
Screen Reader Support	

```
| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The ComboBox control uses the `combobox` role, and each list item has an `option` role. The following ARIA attributes denotes the ComboBox state.

| Properties | Functionalities |

- | --- | --- |
- | `aria-haspopup` | Indicates whether the ComboBox input element has a popup list or not. |
- | `aria-expanded` | Indicates whether the popup list has expanded or not. |
- | `aria-selected` | Indicates the selected option. |
- | `aria-readonly` | Indicates the readonly state of the ComboBox element. |
- | `aria-disabled` | Indicates whether the ComboBox control is in a disabled state or not. |
- | `aria-activedescendent` | This attribute holds the ID of the active list item to focus its descendant child element. |
- | `aria-owns` | This attribute contains the ID of the popup list to indicate popup as a child element. |

| aria-autocomplete | This attribute contains 'both' to a list of options shows and the currently selected suggestion also shows inline. |

Keyboard interaction

You can use the following key shortcuts to access the ComboBox without interruptions.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Selects the first item in the ComboBox when no item is selected. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |

| Enter | Selects the focused item and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the control. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the control. |

| Alt + Down | Open the popup list |

| Alt + Up | Close the popup list |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Cursor moves before the first character in input |

| End | Cursor moves next to the last character in input |

Note: In the following sample, focus the ComboBox control using alt+t keys.

CSHTML

```
@Html.EJS().ComboBox("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Render()
<script>
    document.onkeyup = function (e) {
        if (e.altKey && e.keyCode === 84 /* t */) {
            var dropObj = document.getElementById("games"); //to get
            dropdown list object
            dropObj.ej2_instances[0].focusIn(); // call the focusIn method
            to focus the element.
        }
    };
</script>
```

ACCESSIBILITY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ComboBoxController : Controller
    {
        public ActionResult accessibility()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball",
"Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}

```

Note: [View Sample in GitHub.](#)

Ensuring accessibility

The ComboBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ComboBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ComboBox component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

How To

Autofill supported with ComboBox

The ComboBox supports the **autofill** behaviour with the help of [autofill](#) property. Whenever you change the input value, the ComboBox will autocomplete your data by matching the typed character. Suppose, if there are no matches found then, comboBox doesn't suggest any item.

CSHTML

```

@Html.EJS().ComboBox("games").Autofill(true).Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Fi
elds(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "Name"
}).Render()

```

COUNTRIES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models

```

```
{
    public class Countries
    {
        public string Name { get; set; }
        public string Code { get; set; }
        public List<Countries> CountriesList()
        {
            List<Countries> country = new List<Countries>();
            country.Add(new Countries { Name = "Australia", Code = "AU" });
            country.Add(new Countries { Name = "Bermuda", Code = "BM" });
            country.Add(new Countries { Name = "Canada", Code = "CA" });
            country.Add(new Countries { Name = "Cameroon", Code = "CM" });
            country.Add(new Countries { Name = "Denmark", Code = "DK" });
            country.Add(new Countries { Name = "France", Code = "FR" });
            country.Add(new Countries { Name = "Finland", Code = "FI" });
            country.Add(new Countries { Name = "Germany", Code = "DE" });
            country.Add(new Countries { Name = "Greenland", Code = "GL" });
            country.Add(new Countries { Name = "Hong Kong", Code = "HK" });
            country.Add(new Countries { Name = "India", Code = "IN" });
            country.Add(new Countries { Name = "Italy", Code = "IT" });
            country.Add(new Countries { Name = "Japan", Code = "JP" });
            country.Add(new Countries { Name = "Mexico", Code = "MX" });
            country.Add(new Countries { Name = "Norway", Code = "NO" });
            country.Add(new Countries { Name = "Poland", Code = "PL" });
            country.Add(new Countries { Name = "Switzerland", Code = "CH"
        });
            country.Add(new Countries { Name = "United Kingdom", Code = "GB"
        });
            country.Add(new Countries { Name = "United States", Code = "US"
        });
            return country;
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Configure the Cascading ComboBox

The cascading ComboBox is a series of ComboBox, where the value of one ComboBox depends upon another's value. This can be configured by using the [change](#) event of the parent ComboBox. Within that change event handler, data has to be loaded to the child ComboBox based on the selected value of the parent ComboBox.

The following example shows the cascade behavior of country, state, and city ComboBox. Here, the [dataBind](#) method is used to reflect the property changes immediately to the ComboBox.

CSHTML

```
<div class="col-lg-12 control-section">
    <div class='control-wrapper'>
        <div class="padding-top">
            @Html.EJS().ComboBox("country").Placeholder("Select a
country").PopupHeight("auto").Change("countrychange").DataSource((IEnumerable<Object>
```



```

        ViewBag.country).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text = "CountryName", Value
= "CountryId" }).Render()
        @*//change="countrychange"*@
    </div>
    <div class="padding-top">
        @Html.EJS().ComboBox("state").Placeholder("Select a
state").Enabled(false).Change("statechange").PopupHeight("auto").DataSource(
(IEnumerable<object>)ViewBag.state).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text = "StateName", Value =
"StateId" }).Render()
        @*// change="statechange"*@
    </div>
    <div class="padding-top">
        @Html.EJS().ComboBox("city").Placeholder("Select a
city").Enabled(false).PopupHeight("auto").DataSource((IEnumerable<object>
)ViewBag.cities).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Text = "CityName", Value =
"CityId" }).Render()
    </div>
</div>
</div>
<script>
    function countrychange() {
        var state = document.getElementById('state').ej2_instances[0];
        var city = document.getElementById('city').ej2_instances[0];
        var country =
document.getElementById('country').ej2_instances[0];
        // disable the state DropDownList
        state.enabled = true;
        // frame the query based on selected value in country
        DropDownList.
        var tempQuery = new ej.data.Query().where('CountryId', 'equal',
country.value);
        // set the framed query based on selected value in country
        DropDownList.
        state.query = tempQuery;
        // set null value to state DropDownList text property
        state.text = null;
        // bind the property changes to state DropDownList
        state.dataBind();
        // set null value to city DropDownList text property
        city.text = null;
        // disable the city DropDownList
        city.enabled = false;
        // bind the property changes to City DropDownList
        city.dataBind();
    }
    function statechange() {
        var city = document.getElementById('city').ej2_instances[0];
        var state = document.getElementById('state').ej2_instances[0];
        city.enabled = true;
        // Query the data source based on state DropDownList selected
value
        var tempQuery1 = new ej.data.Query().where('StateId', 'equal',
state.value);

```

```
// set the framed query based on selected value in city
DropDownList.
    city.query = tempQuery1;
    //clear the existing selection
    city.text = null;
    // bind the property change to city DropDownList
    city.dataBind();
}
</script>
```

STATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class State
    {
        public string StateName { get; set; }
        public string CountryId { get; set; }
        public string StateId { get; set; }
        public List<State> StateList()
        {
            List<State> state = new List<State>();
            state.Add(new State() { StateName = "New York", CountryId = "1",
            StateId = "101" });
            state.Add(new State() { StateName = "Queensland", CountryId =
            "2", StateId = "104" });
            state.Add(new State() { StateName = "Tasmania ", CountryId =
            "2", StateId = "105" });
            state.Add(new State() { StateName = "Victoria", CountryId = "2",
            StateId = "106" });
            state.Add(new State() { StateName = "Virginia ", CountryId =
            "1", StateId = "102" });
            state.Add(new State() { StateName = "Washington", CountryId =
            "1", StateId = "103" });
            return state;
        }
    }
}
```

Note: [View Sample in GitHub.](#)

Show the list items with icons

You can render **icons** to the list items by mapping the [iconCss](#) field. This **iconCss** field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with **iconCss** field.

CSHTML

```
@Html.EJS().ComboBox("icons").Placeholder("Select a social
media").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.icondat
```

```
a).Fields(new Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value =
"SocialMediaName", IconCss = "Class" }).Render()
<style>
    @@font-face {
        font-family: 'Socialicons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMvltCfsAAAEoAAAAVmNtYXCnKKeOAAABrAAAAEhnbHlml19
XagAAAgwAABhQaGVhZA8dCeEAAADQAAANmhoZWEIUQQMAAAARAAAACRobXR4LAAAAAAAYAAAA
sbG9jYR3AIwWAAAH0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGx5D00
AABY0AAAAkwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAACwABAAAAQAA+iTiP18
PPPUACwQAAAAANYFYngAAAAAlgVieAAAAAD9AP0AAAACAACAAAAAAAAAAAEAAAAALafQACwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAOkCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCCQAAAAAXAQAAAAAAAAAAAAAAAAAAAAAAAA
EAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAF
EADQAAAAEAAQAAQAPwn//wAApWd//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAAiQ
CzgOMBU4F/gZYB9QICAO+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fHQefDhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLWYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kiW8FAYkFEgQDAyQDAQECAyIBAQMSegkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgwkCQQEBCICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwCFBQUEAwICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRQVFRQV
VDw4ODg0NDAwLCwoKCQkICAcGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwwNDQ0ODg4PDw8QEBA
QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwwNDQ4ODxAREhISEhMTEExMUExQUFRQ
VGxsaGgcIBwFXNgEBAQ8KCgoIBwCGBQUdAwIBAQECAwMDBQUFBgcHCAgICgkKCwsMDAwNDg0ODw8
PEBAQEhISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCACh/aaQEB0cGhgWFBIRDgw
LCACeAwICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQEA8ODhAQDQ0LCQg
HBgQCAgICBAQEBCgYGCAGJCgoLCwwMDQ4ODg8PEBAREBESEhISExITGBcZGRgZGBgXAU4CagMEXAk
FBAQFBCQCAwMGHcKfQkMIwIBAwYAwIBKQECBSgFBgULCgkHBgMBAQEDAwcICgWMDg8PEhITFBu
WGBgSEyYJCAgIBwCHDBEGAgEBAagEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhE
SEREREBEPEA8QDhMEBASFNgEBAQEJCAcGBQMCAQEDAwUGCAgHCAgJCgoKCwsMDA0NDQ4ODg8ODxA
PEA8QEBAQEBAREBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICAcGBgUFAwMDAQEBAQEC6RISExISEhISEhM
SEhESEREREREAQEA8QDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVnCFBAQVEBEREhESEhI
TExMTFBMUehEREBEQEBApDw8PDg4ODQ0MDAwLCwoKCgkICAcHBWUGBQQDAGIBAQEBAGIDBAQFBQc
GCAGICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUFhcYGRscHR8fiiIjFBMTEhMSEhIREhEREBEQEAM
EAwt1YQINCAyEAgIEBAUGBgICQoKDAwNDg8PERUVFhcWGBcYGBkZGRkaGxOTehISEhEREBAQDw8
ODg4MDQwLCgoKCQgHBWYFBQMDAwEBAGMFBQgIAAAAAEAAAAAAzoD9ACWAAATDwYVERUFHTsBPw4
9AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwwMDA0NDQ0ODg4PDw8
QEBDQCGsKCQkJCAgHBWUEBAICAIEBAUHBwgICQkJCgsK0QoKCgoJCAgIBwCFBAMDAQEBKQoJCgg
JCAgHBWUFBAQCAQEBAQIEBAUFBgHCAkJCQkK/tcCagMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgs
KDAwMDf4LExMTEhESEREQEBApDw4PDQ4MDA0KCQgIBgYEBAMCAgEBAwQFBwcJCQoKCwsMDA0NDAs
MCwoKCQkHBWUEAwEBAQEDBAUGCAgJCgoLCwwMDVgCAwMFBgcICQkJCgsLCwwLDAsKCgoJCAgHBgU
EAgIBtQ0MDAsLCgoJCQcHBQQDAQEBAQMEBQYIAAABAAAAAAP0A90AqAAAAT8DMx8MHQEPDSsBLxo
PCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBWUFBBQDAwICAgEDBQoPExYWFAsLCgQFBAUFBAU
FBQUJCQkJCYQEBCUGBwcICAKKCgsLDQwODQ8RERQfI5IvNRMGBWYGBWYGBgYGDAsMWAcICAgICQk
JCQkKCgoLCgsJExITFBQVFRYWFIMkJTEWFRQREQ8NDAsJBWYFAwEBAQEDBAUHBwkJCwwNDhAQEGs
YGBYWFBBQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBCUHBgcICBMSDxEdiiUoIXsOCgcCAQIEBAYICBU
bHyU37xQTERAPDQwKCQgGBQMCAQEDBgLDhofkUInCwIBAgQEBwcJCwscISf+pBYUEXIQDg4LCwk
IBgUDAgEDBACJDA0REhQXJysxRiEhIB4eHRwbGhkZFxcVFRoXfHqTERAODQwKCAcGBAMBAQMFBwk
MDQ8REXUXGhsdAAUAAAAAA/ED9ABCAKoA6wESAYQAAdAQ8NKwEvDjU/EB8OJR0BHw8hPw8TLwM
hHwUVDxEvEzU/ASchDMFFR8PPw8vDw8OAR8HFQ8JIY8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM
9Ai8BIw8EFQ8DIY8CNw8KFxUfASUzPwgZHWkhPwI1LxAlDwICkAMDBQcHCQkKDAwMDQ4ODwwMCws
LCgoJDwsJCAYFAGCAwQGBgcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBgkJDAwODwg
RERITEwJpExMSEhEQDw4MDAUJBWYEAgEBAgEF/uYOCwKGBAICBAYHCQsMDg8QERETExQTFRQVFBQ
UFBMTEhAPDg0LCQgGBAMCAgEBAwMEBQYHCAkC/uoFAGEBASwBAWUGCQoLDQ0PERESEhQUFBQTEhE
QEA4MDAKJBgUDAQEDBQcICgsNDg8QERISFBQUFBMSERAQDgWMCggHBAMCPAYGBgQEAWEBQEBAM
EBAQFBmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFEAKDAwECBAQEBAQDwI
CAwUWAwIBAQQQDwwLCQgFAwEBAQEEATEEBBYUFYRWFxcYFxcXfXyVFBgFBQYBjWYCAgICBAYHCQo
LDA4ODxAIERIR/d8FAQIB9wcHDg0NDAwLCgkIBWYFBAICAQgQEBQYGDQwODg8REBINDQwMCwsKCgk
```

IBwcGBAQCAQEBAgQFBggICgoLDQ0NDg9929sUEXISERAPDgwMBQkHBgQCAQMFBgkJDAwODwgQERI
TEwHBBgMBARYXfxcXFxcWFhUUFbMREQ0ODAsJCAYEAwIBAgQFBwkKDA0PDxERExQPEA8PDw8PDw8
ODw4ODg4OAEBAQKPCgoUEhIREBANDQsKCACFAwEBAwUHCAoLDQ4PEBESExQUFBMTHeEQDw4NCwo
IBwUDAQEDBQcICgsNDg8QEhITEwGSAQICBAUFBgdsBQQFBAQDAgIBAQECAGQFBQYHawcHBgUDAgE
BR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDAwmG0ggEAQICAgTKAQ00EBASEhQ
VEiRdAgIBAQITDg0JCAYDAQFBwomDhQCAQEBAQNNUJBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAAM
AAAAA/QD3QADAFcAlwAANzMRIwUVIzC1IXEzET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQH
W1gIDBQgKCwCHBwgJCQoKCw4NDAsKCAgHBwUEBAICAQHWAQICAgQDBQUFBgYHBwCJCAkJCgoKCws
LDBgZGhQUEREPDg0MCwoJCQ79xAEBawMFBgYHCAkKCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAY
GCAgJCgoLDQwODQ0MDA0KCQkHBwYEBAMBIGKFWwICW/17AXcUDA0ODgWGBQUBAMCAQEBAgMFBQc
ICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQwMCwoKCQkICACGBgUFbAMGAwEBAgMEBgYHCAgICQkSARI
MCwsKCgkICAgGBQUEAwEBAQEDBAUFBgICAKKCgsLDAsLCwsJCggIBWYGBAQDAQEBAQMEBAYGBwg
ICgkLCwsAAAAAABAAAAAPuA/QARgAAExEVHwYhESM1MzU/DzMVIw8GFTMVIxEhPwYRLwYhDwYSAGQ
FBwgKCgHPb24BAwMGBggJCgsMDQ0ODwgPlUcLCwkIBgQDe3sBBQoKCACFBAICBAUHCAoK/IUKCgk
HBwQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBwUEAgGFAwQHCAkKDDOF/mUDBAYICQoLA4I
LCgkIBgQDAQQFBwgKCwAAAAAGAAAAAP0A/QAOABEAIAABBQEqAUwAAAEPCR0Bhw07AT8NPQEvCCM
PASUVMxUjFSM1IzUzNSUPBRUfDTsBPww1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIH1y8NPQE
/DwEVHw8hPw8RITChLw8hDw4BCgMTCwsFBAQEAgICAwQGBgCICgoLDAwODg8NDQwLCgkICAYGBQQ
DAwEBAQIDBAgMDiYRNw0B9nR0TXNz/kAFawMDAQIBAgMDBAQGBgCICQkKDAsIBwCHBwYFBQYFAwM
BAQECAwMEBQYGBwgJCQoLDAcIBwCHBwX+MTAQDggIAwICAQEBAQEDAwMICgsMDAsGAgEBAQECAwY
iGQoFCQcDAgIBAwQFCAGLDA0PERITFRYYFRISEA8NDAsKCAcGBAMCAQEBAwUHCQsOERMUFb0xCAC
DAwEBAQIFGQ4ODQ0LCgoICACFBQQCAGMDBGICGwICBESEhESEBD+pwEDBQYJCgsNDg8IEBISExQ
CahQTEExIREA8ODQsGCQcGBAL8GAED5gIDBgCICgsNDg4QCBIRExP9lhMTExERE40ODQsKCACGAwF
KAQKHCAyGBggICQkKCgkICAgHBgYFBQMDAgIBAgMDBAUFBgYHBwCICQgHBwYGBgYLCwwcBQPYck9
yck5zZwYGBwCHDxELCgWLCwsKCgkJBwUFAwECAwMDBAUHBwCIBw0QCwwLDAsMCgoKCACGBAMBAgI
DAwQFLRkQDwwPCAgJCgoLCQkICAgNDAsKCQwJBQYGBQYEBACbFQsGDA4HCAGJCQ4NDQwNCwwKCgg
IBgYDAwEBAgMDBQYGBwgJCAoJCgsKCwUMDAwMDAsKCQYFBQUKDAYHCAgJBw0BAgQEBQcHCAkJCgo
KCwsLDQ4NDQ0MDAsGBgkIBQQCAQH+EAoKExMSERAQDQ0LBgkHBgQCAQMFBgkKCw0NEAgQEhITFAJ
HKwQSEhIQDw8NDAsJBQcFBAIBAwQHCAkLDA0PDxASEhIAAAAAAgAAAAAD7gP0AEAaHAAAAARUzFSM
RHws/BxUPAy8OESM1Pw81ER8OMyEzPw4RLw4jISMPDQIbysoDBgUICgYHCAgJCgsLDQ4PEBESE0Q
tICIiEREQDw8ODQwKCgCHBANuGBkVDw4ODgYFBQEBAwMCAv5fAQECAwQEBQUGBwCHCAgJCAM0CAk
ICACHBwYFBQQEAWIBAQEBAgMEBAUFBgCHBwgICQj8zAgJCAgHBwCGBQUBAMCAQON0H/+9BIMCAk
HBAMDAgEBAQEBAgMDBQYHeA4GAwEBAgIDBAUFBgJCwsNDxABVGwKDXANDxEUCwwMDQ00DxAQEvz
CCQgICACHBwYGBAUDAwICAgIDAUEBgYHBwCICAgJAz4JCAgIBwCHBwYEBQMDAgICAgMDBQQGBgC
HBwgICAAAAGAAAAAD7APzAPgBqAAAR8LFQ8MIY8QKwEPDh8bHQEPfi8WPQE/DTMfEjM/Di8ePQE
/Fh8CBR8HDwMfhjSBPwIfBzM/HTUvBz8CPQEvHiMPai8HIw8dAnALFhMSDw4LCQgFBAIBAgIDAww
FBgUGBgCGCAwLCQgHChQLCwsHBwkJCgsNDQwMCwsJCggIBWYFBAMDAQEBAgMEBQcHCRMTdxojFhQ
TEA8OCwUFAwQCAwEBAgIEBQUHCAgKCgWMDg4PEBERehMTFBUZGBYWFRMSEgsLCwoJCQgIBWYFBQM
CAgECAgMDBAUFBQYGBgYHCAAsLCgkIBwCMBwCHBwKDAcPERMZDQ0MDAsKCQgHBgUEAwEBAQICAgM
EBAsMDQ8bTSIfGxkMCwsKCQgIBWYFBQMCAgICBAQGBgCICQoLDA0NDw8PERERExIUHxwb/bsBAgM
EBQcHCQUADAQEBAQMFBQYICAKLCwwNDg8QEBESEhMUFBWFRcWGBcYGBYWFRUPDxAQEBEREQ4ODg0
NDQ0MDAwMCwoLCgkJCQgHBwCGBgQFAwMDAgEBAQIDBAUGBgQEAgIDBAUHBwkJCgWMDQ4PDxERERM
TFBQVFRYWFxcYGBgUFRQTEBESEhITFBMODg4NDQ0NDA0LDAsKCwoJCQkIBwCHBwYEBQMDAwIBAzC
ECAoLDAwNDQ4NDg0NBgYGBQYKBQQDAwICAQECBAUHDSEODQoEBAMCAgIBAQICAwMEBQUFBQYGBgY
GCAcHBgYFBQYIBx0GDAgJCgsNDg8JCAkKCgsLCwwPDg0ODQwMDAsLCgoICAgHBgUEBAMCAQEBAgI
EBQYICAYIBwkJCQoKCwsLCgsLCgoHBgYGBQUFBQQAEMCAQEBAgUGCAkLGg0LCgkICAYDBAMCAQI
DBAQFBgYGBwCIBwkIDQcFBgUEBQgIBgYHEgkJCgoHBgCICAKJCgoLDAwMDg0NDQ0MDAsLCgoKCQg
IBWYGBQGEAwMBAQEBAwRbEhMSEREREBAXfXgYBgYfXcWFhUVFBQTEExEREQ8PDg0MDA0KCACHBQ
DAGICAwCGBgUDAwEBAQIDAwmFBAyGBwCHCAkJCQoLDA0NDw8MDAwMDQ0NDQ4ODHAQE8PDw4OGB0ZGhg
YfXgWfXUWFRQUExISERAQDw4NDAsLCQkHBgUFAwEBAgIDCggHBgUDAgEBAgMDAwUEBgYGCACICQk
JCgoLCwwLDQwNDQ0NDg4AAAAACwAAAAAD8wOYABEAMwBbAKYAYwDTARcBOQFjAZGBoQAAAQ8DMzc
vBisBDwEnDwIdAh8FOWE/BjUvBisBDwEnFwcfBDM/Bic1MxUnNw8GIY8HNyClHwsVIxUfBjsBPwY
1MxcVDw0vCzU/CycVPwMfCR0CDwgjLwQPAREjFSMVIzUjNTcPCxUfDyE/DzUvDiMhIw8BJR8DFQ8
GKwEvBjU/Bx8DFR8KPwUHMzUjFQ8GKwEvBjUjDwcdAR8LowE/CTUvDg8DFTM1NyMHJyMDIgQDAGJ
CAgECAwQFBQYGCgUG2QQDAgIDBAUFbQYGBQYEBQICAQECAgUEBgUGBgUF6wEBAwUCAwMEBwGEAgE
BAQFFOQEDBAYMDhAQDwgGBgYfAgIEAQECHQoLCgkJCAcFBAMCAXcBAQMDBQQFBhAGBQQEAgIBMwM
BAQMCBAQMBwCICAgIDxAPDg8HBgYEAwMBAQECAgQGBggJCQkJC+UQDg4NDQUGBwCFBAQCAgIDAuU

```

GBgcICQoLBQwNFAQ50VJFRyAPDQ0LCwkJBwMFAwIBAgQGBwkJCwsNDQ8PDxARAqAQEQ8PDw0NCws
KCAcDBQMCAQIEBgcICgsLDQ0PDxAQEP1gERAPAYoEAgIBAgICBAUGBQYGBgUFBAICAQECaWQFBQU
GBgUGdgEEAgQEBQQGBWgIBwcGBgoKAUw7AQEDAwQEBQUFBQQEAWIBAUC+CggGBgMCAgICBAMECgY
HCQsLCwwMCwoSBwcHCAUEAgEBAgIEAwQHBWgJCgsMDg8NDMPKV1AuLVEBbAMEBB8bBQMEAwICAQE
CCAMDawOAAwMDAgICAQECAGIDAwoAAwMDAwICAQECF4kgBQUCAQECBAQEAQIDCJrYARsEBAMHBQQ
CAQICAwQEBRoPmw0BAQIDAQWQFBgYMDBGvMgYEAwIDAQEBaQMCAwQEGREGBgUFBBQQECQQEAwICAQE
BAwYHBQYGBWgJCgPDDwWLCggHBgYDAwMBQFQHBQQAQICAwUFBQYHBwhwCgkJCAGBQQDAQEBAU
LEgEBIib+/yVJBQUGBwcICQkFCgsK9gsLCgoJCQgHBWYFBQMDAQEBaQMDBQUGBwcICQkFCgoL9wo
LCgoJCQgHBWYFBQMDAgID9wQEBaV3BAUDbAMCAQECaWQEBAR3BQQEBAMCAQEBaQJ3GAwRBAUEAwM
DAQEBaQEBaWcKEuCvAwMDAgMBAQEBaWIDAwoVAgUHCAoKDA0OQxgOBwcGCgQEAgMCAQICBQQFBws
KDxZTCgkHBgYGBQYFBQMDAgEBAQIEPayslG9vAAAAABIA3gABAAAAAEEEEAAABAAAAAABAAAs
AAQABAAAAAACAACADAABAAAAAADAAsAEwABAAAAAEEAAsAHgABAAAAAFAAsAKQABAAAAAAs
GAASANAABAAAAAACAkCwAPwABAAAAAALABIAawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAE
ECQACAA4AlQADAAEECQADABYAowADAAEECQAEABYAuQADAAEECQAFABYazwADAAEECQAGABYA5QA
DAAEECQAKAFgA+wADAAEECQALACQBUyBTb2NpYWxpY29uc1JlZ3VsYXJlZ3VsYXJlZ3VsYXJlZ3Vs
hbG1jb25zVmVyc2lvbiAxLjBTb2NpYWxpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAUwBvAGMAaQBhAGwAaQBjAG8AbgB
zAFIAZQBnAHUAbABhAHIAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAFMAbwBjAGkAYQBsAGkAYwBvAG4
AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAEYAbwBuAHQAIAb
nAGUAbgBlAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAEQBuAGMAZgBlAHMAaQBvAG4AIABNAGU
AdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQA
AAAACAAAAAAsAAAAAAsAAAAAAsAAAAAAsAAAAAAsBAgEDAQQBBQEGAQcBCAEJAQoBCwE
MAAh3aGf0c2FwcAd0d2l0dGVyBXZpbWVvCWluc3RhZ3JhbQhsaW5rZWRpbgYWNlYm9vawtnb29
nbGUtcGxlZ3d0dWl1bHIIc2t5cGUtMDEIeW9ldHViZTEAAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}
.e-list-icon {
font-family: 'Socialicons' !important;
color: rgba(0, 0, 0, .57);
}
.twitter:before {
content: "\a701";
}
.vimeo:before {
content: "\a702";
}
.youtube:before {
content: "\a709";
}
.whatsapp:before {
content: "\a700";
}
.skype:before {
content: "\a708";
}
.instagram:before {
content: "\a703";
}
.google-plus:before {
content: "\a706";
}
.facebook:before {
content: "\a705";
}
.tumblr:before {
content: "\a707";
}

```

```

    }
    .linkedin:before {
        content: "\a704";
    }
</style>

```

SOCIALMEDIA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class SocialMedia
    {
        public string Class { get; set; }
        public string SocialMediaName { get; set; }
        public string Id { get; set; }
        public List<SocialMedia> SocialMediaList()
        {
            List<SocialMedia> media = new List<SocialMedia>();
            media.Add(new SocialMedia { Class = "facebook", SocialMediaName
= "Facebook", Id = "media1" });
            media.Add(new SocialMedia { Class = "google-plus",
SocialMediaName = "Google Plus", Id = "media2" });
            media.Add(new SocialMedia { Class = "instagram", SocialMediaName
= "Instagram", Id = "media3" });
            media.Add(new SocialMedia { Class = "linkedin", SocialMediaName
= "LinkedIn", Id = "media4" });
            media.Add(new SocialMedia { Class = "skype", SocialMediaName =
"Skype", Id = "media5" });
            media.Add(new SocialMedia { Class = "tumblr", SocialMediaName =
"Tumblr", Id = "media6" });
            media.Add(new SocialMedia { Class = "twitter", SocialMediaName =
"Twitter", Id = "media7" });
            media.Add(new SocialMedia { Class = "vimeo", SocialMediaName =
"Vimeo", Id = "media8" });
            media.Add(new SocialMedia { Class = "whatsapp", SocialMediaName
= "WhatsApp", Id = "media9" });
            media.Add(new SocialMedia { Class = "youtube", SocialMediaName =
"YouTube", Id = "media10" });
            return media;
        }
    }
}

```

Note: [View Sample in GitHub.](#)

ComboBoxFor

The ComboBoxFor component can be rendered by passing values and data from the model. The selected values can be retrieved during form submit using the post method.

CSHTML

```
@using (Html.BeginForm())
```

```
{
    @Html.EJS().ComboBoxFor(model => model.Value).Placeholder("Select a
Country").DataSource((IEnumerable<object>)ViewBag.data).Fields(new
Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "Name" }).Render()
    <div id="submitButton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}
```

FOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        Countries model = new Countries();
        public ActionResult Index()
        {
            ViewBag.data = new Countries().CountriesList();
            model.Value = "Cameroon";
            return View(model);
        }
        [HttpPost]
        public ActionResult Index(Countries model)
        {
            ViewBag.data = new Countries().CountriesList();
            model.Value = model.Value;
            return View();
        }
    }
}
```

Data Annotation

Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end users.

Data Annotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET MVC Framework will automatically enforce these validation rules and display validation messages in the view.

Using **value** property gets or sets the value of the selected item in the component.

CSHTML

```
@using (Html.BeginForm())
{
    @Html.EJS().ComboBoxFor(model =>
model.EnquiringAboutSelect).Placeholder("Select a
Country").DataSource(Model?.EnquiringAboutSelectListItems).Fields(new
```

```

Syncfusion.EJ2.DropDowns.ComboBoxFieldSettings { Value = "Text"
}).Placeholder("Please Select Enquiring About").Render()
    <div>
        @Html.ValidationMessageFor(model => model.EnquiringAboutSelect)
    </div>
    <div id="submitButton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}

```

FOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {
            DataModel model = new DataModel();
            model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
            model.EnquiringAboutSelect = "Beacon";
            return View(model);
        }
        [HttpPost]
        public ActionResult Index(DataModel model)
        {
            model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
            model.EnquiringAboutSelect = model.EnquiringAboutSelect;
            return View(model);
        }
    }
    public class DataModel
    {
        [Required(ErrorMessage = "The value is Required")]
        public string EnquiringAboutSelect { get; set; }
        public List<ListItems> EnquiringAboutSelectListItems { get; set; }
    }
    public class ListItems
    {
        public string Text { get; set; }
        public string Value { get; set; }
        public List<ListItems> getListItems()
        {
            List<ListItems> items = new List<ListItems>();
            items.Add(new ListItems() { Text = "Aberdeen", Value = "103" });
            items.Add(new ListItems() { Text = "Alexandria", Value = "102"
});
            items.Add(new ListItems() { Text = "Albany", Value = "101" });

```



```

        items.Add(new ListItems() { Text = "Beacon ", Value = "104" });
        items.Add(new ListItems() { Text = "Brisbane ", Value = "105"
    });
    return items;
}
}
}

```

Migration from Essential JS 1

This article describes the API migration process of ComboBox component from Essential JS 1 to Essential JS 2.

DataBinding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:**

Datasource
Html.EJ().ComboBox("select").Datasource((IEnumerable<CarsList>)ViewBag.datasources).Render() | **Property:**

DataSource
@Html.EJS().ComboBox("games").DataSource((IEnumerable<object>)ViewBag.localdata).Render() |

| **Fields for mapping** | **Property:**

ComboBoxFields
Html.EJ().ComboBox("select").ComboBoxFields(f=>f.Text("text")).Render() |

Property: *Fields*
@Html.EJS().ComboBox("games").Fields(new ComboBoxFieldSettings { Text = "Game", Value = "Id" }).Render() |

| **Query** | **Property:**

Query
Html.EJ().ComboBox("select").Query("ej.Query().from('Suppliers').select('SupplierID', 'ContactName')").Render() | **Property:**

Query
@Html.EJS().ComboBox("games").Query((string)ViewBag.query).Render() |

| **Begin event** |

Event: *ActionBegin*
Html.EJ().ComboBox("select").ActionBegin("onBegin").Render() | **Event:**

ActionBegin
Html.EJ().ComboBox("select").ActionBegin("onBegin").Render() |

| **Complete event** |

Event: *ActionComplete*
Html.EJS().ComboBox("select").ActionComplete("onComplete").Render() | **Event:**

ActionComplete
Html.EJS().ComboBox("select").ActionComplete("onComplete").Render() |

| **Failure event**

Event: *ActionFailure*
Html.EJ().ComboBox("select").ActionFailure("onFailure").Render() |

Event: *ActionFailure*
Html.EJS().ComboBox("select").ActionFailure("onFailure").Render() |

Filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | Property:

AllowFiltering
`Html.EJ().ComboBox("select").AllowFiltering(true).Render()` | **Property:**
`AllowFiltering`
`Html.EJS().ComboBox("select").AllowFiltering(true).Render()` |

| No records template | Property:

NoRecordsTemplate
`Html.EJ().ComboBox("select").NoRecordsTemplate(" NO DATA AVAILABLE").Render()` | **Property:**
`NoRecordsTemplate`
`Html.EJS().ComboBox("select").NoRecordsTemplate(" NO DATA AVAILABLE").Render()` |

| Ignore casing and diacritics | Not Applicable | Property:

IgnoreAccent
`Html.EJS().ComboBox("select").AllowFiltering(true).Render()` |

| Custom value addition | Property:

AllowCustom
`Html.EJ().ComboBox("select").AllowCustom(true).Render()` |
<https://ej2.syncfusion.com/aspnetmvc/ComboBox/CustomValue#/material> |

| Search event | Event: Filtering
`Html.EJ().ComboBox("select").Filtering("Filtering").Render()`

| Event: Filtering
`Html.EJS().ComboBox("select").Filtering("Filtering").Render()` |

Template

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | Property: *ItemTemplate*
`Html.EJ().ComboBox("select").ItemTemplate("<div><div class='ename'> ${text} </div><div class='temp'> ${country} </div></div>").Render()` | **Property:**
`ItemTemplate`
`Html.EJS().ComboBox("select").ItemTemplate("<div><div class='ename'> ${text} </div><div class='temp'> ${country} </div></div>").Render()` |

| Group Template | Property:

GroupTemplate
`Html.EJ().ComboBox("select").GroupTemplate("${country}").Render()` | **Property:**
`GroupTemplate`
`Html.EJS().ComboBox("select").GroupTemplate("${country}").Render()` |

| ValueTemplate | Not Applicable | Property:

ValueTemplate
`Html.EJS().ComboBox("select").ValueTemplate(" NO DATA AVAILABLE").Render()` |

| Header Template | Property:

HeaderTemplate
`Html.EJ().ComboBox("select").HeaderTemplate("<div class='head'> Photo Contact Info </div>").Render()` | **Property:**
`HeaderTemplate`
`Html.EJS().ComboBox("select").HeaderTemplate("<div class='head'> Photo Contact Info </div>").Render()` |

| FooterTemplate | Property:

FooterTemplate
`Html.EJ().ComboBox("select").FooterTemplate("<div class='Foot'> Total Items Count: 5 </div>").Render()` | **Property:**

FooterTemplate
Html.EJS().ComboBox("select").FooterTemplate("<div class='Foot'> Total Items Count: 5 </div>").Render() |

| **No records Template** | **Property:**

NoRecordsTemplate
Html.EJ().ComboBox("select").NoRecordsTemplate(" NO DATA AVAILABLE").Render() | **Property:**

NoRecordsTemplate
Html.EJS().ComboBox("select").NoRecordsTemplate(" NO DATA AVAILABLE").Render() |

| **Auto fill** | **Property:** *AutoFill*
Html.EJ().ComboBox("select").AutoFill(true).Render() | **Property:**

AutoFill
Html.EJS().ComboBox("select").AutoFill(true).Render() |

| **Action failure Template** | **Property:**

ActionFailureTemplate
Html.EJ().ComboBox("select").ActionFailureTemplate("Data fetch get fails").Render() | **Property:**

ActionFailureTemplate
Html.EJS().ComboBox("select").ActionFailureTemplate("Data fetch get fails").Render() |

Applying CSS

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:** *CssClass*

CssClass
Html.EJ().ComboBox("select").CssClass("customclass").Render() | **Property:**

CssClass
Html.EJS().ComboBox("select").CssClass("customclass").Render() |

| **width** | **Property:** *Width*
Html.EJ().ComboBox("select").Width("300px").Render() |

Property: *Width*
Html.EJS().ComboBox("select").Width("300px").Render() |

Grouping

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:**

ComboBoxFields
Html.EJ().ComboBox("select").ComboBoxFields(f=>f.GroupBy("text")).Render() | **Property:** *Fields*
@Html.EJS().ComboBox("games").Fields(new ComboBoxFieldSettings { GroupBy = "Game" }).Render() |

Accessibility

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Globalizaation** | **Property:** *Locale*
Html.EJ().ComboBox("select").Locale("fr-FE").Render() |

Property: *Locale*
Html.EJS().ComboBox("select").Locale("fr-FE").Render() |

| **Rtl support** | **Property:**

EnableRtl
Html.EJ().ComboBox("select").EnableRtl(true).Render() | **Property:**

EnableRtl
Html.EJS().ComboBox("select").EnableRtl(true).Render() |

Placeholder

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Watermark text** | **Property:**

Placeholder
Html.EJ().ComboBox("select").Placeholder("Select").Render() | **Property:**
Placeholder
Html.EJS().ComboBox("select").Placeholder("Select").Render() |

| **Floating of watermarkText** | **Not applicable** | **Property:**

FloatLabelType
Html.EJS().ComboBox("select").FloatLabelType(FloatLabelType
.Auto).Render() |

Miscellaneous

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Enable/disable** | **Property:**

Enabled
Html.EJ().ComboBox("select").Enabled(true).Render() | **Property:**
Enabled
Html.EJS().ComboBox("select").Enabled(true).Render() |

| **Read only** | **Property:** *ReadOnly*
Html.EJ().ComboBox("select").ReadOnly(true).Render()

| **Property:** *ReadOnly*
Html.EJS().ComboBox("select").ReadOnly(true).Render() |

| **Addition of Html attributes** | **Property:**

HtmlAttributes
Html.EJ().ComboBox("select").HtmlAttributes(ViewBag.attr).Render() | **Property:**
HtmlAttributes
Html.EJS().ComboBox("select").HtmlAttributes(ViewBag.attr).Render() |

Sorting

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Order of sorting** | **Property:**

SortOrder
Html.EJ().ComboBox("select").SortOrder(SortOrder.Ascending).Render() | **Property:**
SortOrder
Html.EJS().ComboBox("select").SortOrder(SortOrder.Ascending).Render() |

Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Selecting particular index** | **Property:** *Index*
Html.EJ().ComboBox("select").Index(1).Render()

| **Property:** *Index*
Html.EJS().ComboBox("select").Index(1).Render() |

| **Selecting particular value** | **Property:**

Value
Html.EJ().ComboBox("select").Value("Car").Render() | **Property:**
Value
Html.EJS().ComboBox("select").Value("Car").Render() |

| **Selecting particular text** | **Property:** *Text*
Html.EJ().ComboBox("select").Text("Car").Render()

| **Property:** *Text*
Html.EJS().ComboBox("select").Text("Car").Render() |

| **Getting data by using value** | **Method:**

getItemDataByValue
Html.EJ().ComboBox("dropdown").Render()

\$('#dropdown').ejDropDownList('getItemDataByValue','data') | **Method:**

getDataByValue
`Html.EJ().ComboBox("combobox").Render()`
`var cmbObj = document.getElementById(combobox).ej2_Instances[0];`
`cmbObj.getDataByValue("data");` |
Select event | **Event:** *Select*
`Html.EJ().ComboBox("dropdown").Select("onSelect").Render()` |
Event: *Select*
`Html.EJS().ComboBox("dropdown").Select("onSelect").Render()` |

Popup

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

Popup height | **Property:**

PopupHeight
`Html.EJ().ComboBox("dropdown").PopupHeight("300px").Render()` | **Event:** *PopupHeight*
`Html.EJS().ComboBox("dropdown").PopupHeight("300px").Render()` |

Popup width | **Property:**

PopupWidth
`Html.EJ().ComboBox("dropdown").PopupWidth("300px").Render()` | **Event:** *PopupWidth*
`Html.EJS().ComboBox("dropdown").PopupWidth("300px").Render()` |

Popup showing manually | **Method:** *showPopup*
`Html.EJ().ComboBox("dropdown").Render()`
`

$('#dropdown').ejComboBox("showPopup");` | **Method:**
showPopup
`Html.EJS().ComboBox("combobox").Render()`
`var cmbObj = document.getElementById(combobox).ej2_Instances[0];`
`cmbObj.showPopup();` |

Popup hiding manually | **Method:** *hidePopup*
`Html.EJ().ComboBox("dropdown").Render()`
`

$('#dropdown').ejComboBox("hidePopup");` | **Method:**
hidePopup
`Html.EJS().ComboBox("combobox").Render()`
`var cmbObj = document.getElementById(combobox).ej2_Instances[0];`
`cmbObj.hidePopup();` |

Popup hide event | **Event:**

Close
`Html.EJ().ComboBox("dropdown").Close("onClose").Render()` | **Event:**
Close
`Html.EJS().ComboBox("dropdown").Close("onClose").Render()` |

Popup shown event | **Event:**

Open
`Html.EJ().ComboBox("dropdown").Open("onOpen").Render()` | **Event:**
Open
`Html.EJS().ComboBox("dropdown").Open("onOpen").Render()` |

Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

Adding new item | **Method :** *addItem*
`Html.EJ().ComboBox("dropdown").Render()`
`
$('#dropdown').ejComboBox("addItem", { text : "India" });` | **Method:**
addItem
`Html.EJ().ComboBox("combobox").Render()`
`var cmbObj = document.getElementById(combobox).ej2_Instances[0];`
`cmbObj.addItem({Id: 'id', Game: 'Golf'},2);` |

Focus out event | **Not applicable** | **Event:**

Blur
`Html.EJS().ComboBox("dropdown").Blur("onBlur").Render()` |

Focus in event | **Event:** *Focus*
`Html.EJ().ComboBox("dropdown").Focus("onFocus").Render()` |

Event: *FocusIn*
`Html.EJS().ComboBox("dropdown").Focus("onFocus").Render()` |

| **Focus out** | **Method:** `focusOut
Html.EJ().ComboBox("dropdown").Render()

$('#dropdown').ejComboBox("focusOut");` | **Method:**
`focusOut
Html.EJS().ComboBox("combobox").Render()

var cmbObj =
document.getElementById(combobox).ej2_Instances[0];

 cmbObj.focusOut();` |

| **Focus in** | **Method:** `focusIn
Html.EJ().ComboBox("dropdown").Render()

$('#dropdown').ejComboBox("focusIn");` | **Method:**
`focusIn
Html.EJS().ComboBox("combobox").Render()

var cmbObj =
document.getElementById(combobox).ej2_Instances[0];

 cmbObj.focusIn();` |

| **Getting the data** | **Method :** `getItems
Html.EJ().ComboBox("dropdown").Render()

$('#dropdown').ejComboBox("getItems");` | **Method:**
`getItems
Html.EJS().ComboBox("combobox").Render()

var cmbObj =
document.getElementById(combobox).ej2_Instances[0];

 cmbObj.getItems();` |

| **Create event** | **Event:**
`Create
Html.EJ().ComboBox("dropdown").Create("onCreate").Render() | Event:
Created
Html.EJS().ComboBox("dropdown").Created("onCreate").Render() |`

| **Change event** | **Event:**
`Change
Html.EJ().ComboBox("dropdown").Change("onChange").Render() | Event:
Change
Html.EJS().ComboBox("dropdown").Change("onChange").Render() |`

| **Custom value event** | **Event:** `custom-value-
specifier
Html.EJ().ComboBox("dropdown").CustomValueSpecifier("fucntion").Render() |`
Event:
`CustomValueSpecifier
Html.EJS().ComboBox("dropdown").CustomValueSpecifier("fucntion").
Render() |`

ContextMenu

Getting Started with ASP.NET MVC ContextMenu Control

This section briefly explains about how to include [ASP.NET MVC ContextMenu](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```


Add ASP.NET MVC ContextMenu control

Now, add the Syncfusion ASP.NET MVC ContextMenu control in `~/Views/Home/Index.cshtml` page.

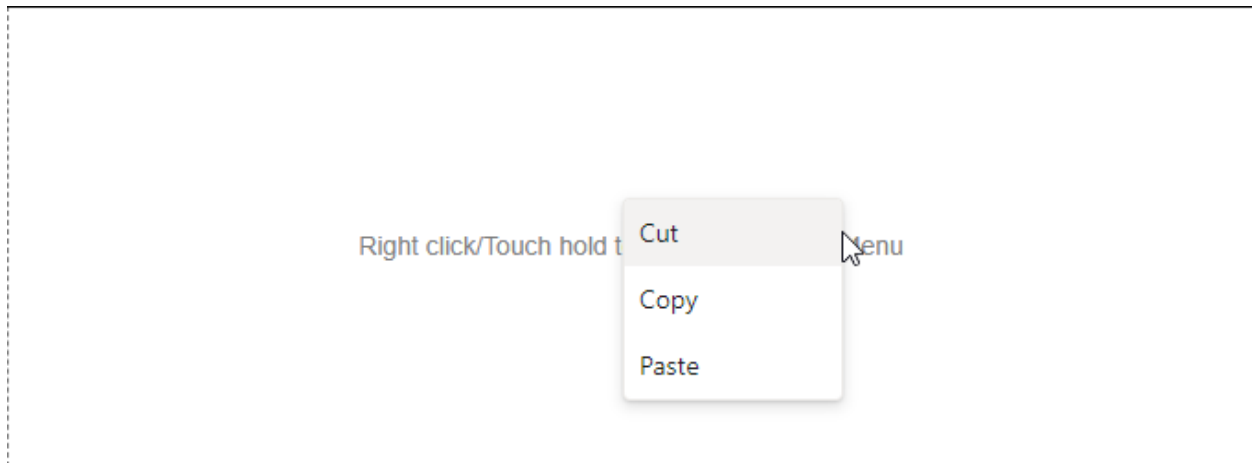
CSHTML

```
@model List<object>
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)Model).Render()
<style>
    #contextmenutarget {
        border: 1px dashed;
        height: 250px;
        padding: 10px;
        position: relative;
        text-align: center;
        color: gray;
        line-height: 17;
        font-size: 14px;
    }
</style>
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "Cut"
    });
    menuItems.Add(new
    {
        text = "Copy"
    });
    menuItems.Add(new
    {
        text = "Paste"
    });
    return View(menuItems);
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC ContextMenu control will be rendered in the default web browser.



Rendering items with Separator

The Separators are the horizontal lines that are used to separate the menu items. You cannot select the separators. You can enable separators to group the menu items using the [Separator](#) property. Cut, Copy, and Paste menu items are grouped using the `separator` property in the following sample.

CSHTML

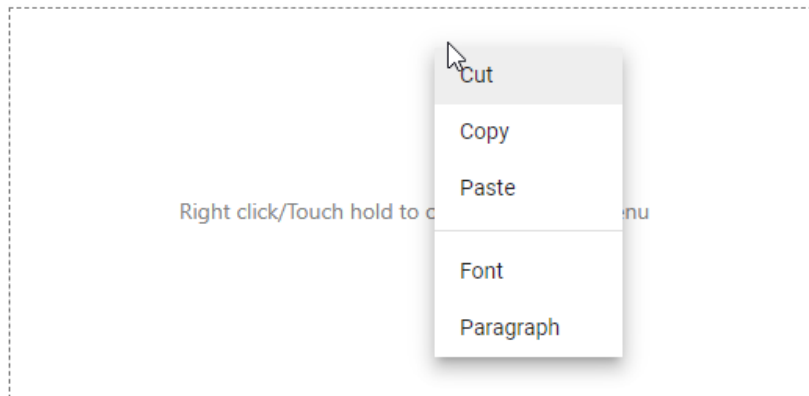
```
@model List<object>
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)Model).Render()
<style>
    #contextmenutarget {
        border: 1px dashed;
        height: 250px;
        padding: 10px;
        position: relative;
        text-align: center;
        color: gray;
        line-height: 17;
        font-size: 14px;
    }
</style>
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    //define the array of JSON
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "Cut"
    });
    menuItems.Add(new
    {
        text = "Copy"
    });
    menuItems.Add(new
```

```
{
    text = "Paste"
});
menuItems.Add(new
{
    separator = true
});
menuItems.Add(new
{
    text = "Font"
});
menuItems.Add(new
{
    text = "Paragraph"
});
return View(menuItems);
}
```

Note: The [Separator](#) property should not be given along with the other fields in the [MenuItem](#).



Note: [View Sample in GitHub](#).

See also

- [ContextMenu with icons](#)
- [Multi-level nesting](#)

Icons and Navigation

Icons

The ContextMenu item has an icon or image in it to provide visual representation of the action. To place the icon on a menu item, set the [IconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the menu item. In the following sample, the icons for Cut, Copy and Paste menu items are added using the [IconCss](#) property. (<https://help.syncfusion.com/cr/aspnetmvc-js2/Syncfusion.EJ2.Navigations.ContextMenuItem.html#SyncfusionEJ2NavigationsContextMenuItemIconCss>) property.

CSHTML

```

<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((I
Enumerable<Object>) ViewBag.items).Render()

<style>
#target {
    border: 1px dashed;
    height: 150px;
    padding: 10px;
    position: relative;
    text-align: justify;
    color: gray;
    user-select: none;
}
/* csslint ignore:start */
@@font-face {
    font-family: 'ddb-icons';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAVMnTYXDnE+dkAAABlAAAADxnbH1mlh3
3NQAAAdwAAAJMaGVhZBKOK9sAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCGAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADYgABAAAAAAAAAAAAAAAAABQABAAAAQAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAAACAACAAAAAAAAAAAAEAAAFaFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAAEAAgADAAQAAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYYAAALHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMyYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEffXo6IBkNCQIHcy8bCQG9BwIJDRkgOhoXHwoKgi/+TRlRDyE
OIxo+ExckFAQMfIkVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsakQ4ONzcniiYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDDQAEAAAAAAOqA+kABQANABcAHwAAARUzFSErAyERIZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAAPKA+oAAGATAAABFxEbDgEHHgEXETMRMxE
zETMlIQl+zPlabpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAAQAAAAAAQAJAEAAQAAAAAAAGAHAAoAAQAAAAAAAwAJABEAAQAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAACgAsAdcAAQAAAAAACwASAGMAAwABBakAAAACAHU
AAwABBakAAQASAHcAAwABBakAAgAOAIkAAwABBakAAwASAJcAAwABBakABAASAKkAAwABBakABQA
WALsAAwABBakABgASANEAAwABBakACgBYAOMAawABBakAcWakATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlcn2l2lbiBNZXRybyBTdHVkaW93d3cuc3luY2Zlcn2l2lbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwBlAGwAYQBYAGQAZABiAC0AaQBJAG8AbgBzAGQAZABiAC0AaQBJAG8AbgBzAFY
AZQBYAHMAaQBVAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIABlAHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxXzBGZvbnQ
OcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
/* csslint ignore:stop */
.e-cm-icons {
    font-family: 'ddb-icons' !important;
    speak: none;

```

```
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-cut::before {
    content: '\e700';
}
.e-copy::before {
    content: '\e70a';
}
.e-paste::before {
    content: '\e701';
}
</style>
```

ICONS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public IActionResult Icons()
        {
            List<object> items = new List<object>();
            items.Add(new
            {
                text = "Cut",
                iconCss = "e-cm-icons e-cut"
            });
            items.Add(new
            {
                text = "Copy",
                iconCss = "e-icons e-copy"
            });
            items.Add(new
            {
                text = "Paste",
                iconCss = "e-cm-icons e-paste"
            });
            ViewBag.items = items;
            return View();
        }
    }
}
```

Note: The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icons on the contextmenu menuitems using the [IconCss](#) property.

Navigation

Navigation in ContextMenu is used to navigate to the other web page when menu item is clicked. This can be achieved by providing link to the menu item using the [url](#) property. In the following sample, Navigation URL for Flipkart, Amazon, and Snapdeal menu items are added using the `url` property.

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)ViewBag.items).Render()
<style>
#target {
    border: 1px dashed;
    height: 150px;
    padding: 10px;
    position: relative;
    text-align: justify;
    color: gray;
    user-select: none;
}
/* csslint ignore:start */
@@font-face {
    font-family: 'cart';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj0gSQ4AAAEoAAAAMvNtYXNlE0dVAAABiAAAADZnbHlmGat
ngwAAAcgAAADYAGVhZBktP4wAAADQAAANmhoZWEHmQNpAAAArAAAACRobXR4B+j//gAAAYAAAA
IbG9jYQBsAAAAAHAAAAABmlheHABDwBQAAABCAAAACBuYw1lfiv21QAAAgAAAAIBcG9zdIZzcJA
AAASkAAAAOgABAAADUv9qAFoEAP/+//wD7AABAAAAAAAAAAAAAAAAAAAgABAAAAQAA2UwSaF8
PPPUACwPoAAAAANfSfWUAAAAA19J9Zf/+AAAD7APdAAAAACAACAAAAAAAAAAEAAAACAEQAawAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQP0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPdAJYAAAABAAAAAAAAABAAAAAPo//4
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAEAAAA
AAAAAbAAAAAP//gAAA+wD3QAJABIAQwAAAJR4BMjY0JicOAQUeATI2NCYiBgEOAwclIgyXEX4BMwU
yFgcOAQclIgyXBhYXBT4BPwE2PwI2Nxm+AycmIyIGAeABJzonJx0gJ/6jASc6Jyc6JwMXIDgZPyX
9giQkBkIIQyQBACQgCQo/JP7RIxcBARcjAVgkQg0QDgkICgkNkw4xMBwCBScJE1UdJyc6JwEBJx0
dJyc6JycDZQg0PigBAY0k/uAkMAQnHRsmAQMTDA8QAQMBLCilIhYWFxYhAYciNg4YDBMCAAAAEgD
eAAEAAAAAAAAAAQAAAAEAAAAAAAAEABAABAAEAAAAAAAAIABwAFAAEAAAAAAAAAMABAAMAAEAAAAAAQ
ABAAQAAEAAAAAAAAUACwAUAAEAAAAAAAAAYABAAfAAEAAAAAAAAoALAAjAAEAAAAAAAsAEgBPAAMAAQQ
JAAAAAgBhAAMAAQQJAEEACABjAAMAAQQJAAIADgBrAAMAAQQJAAMACAB5AAMAAQQJAAQACACBAAM
AAQQJAAUAFgCJAAMAAQQJAAAYACACfAAMAAQQJAAoAWACnAAMAAQQJAAAsAJAD/IGNhcnRSZWdlbGF
yY2FydGNhcnRWZXJzaW9uIDEuMGNhcnRGb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1
ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGMAYQByAHQAUGBlAGcAdQBsaGEAcgBjAGE
AcgB0AGMAYQByAHQAVgBlAHIAcWBPAG8AbgAgADEALgAwAGMAYQByAHQARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAADQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMADABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwBtAAAAAII
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMAEHNob3BwaW5nLWNhcnQtMDUAAAAA
A) format('trueType');
    font-weight: normal;
    font-style: normal;
}
/* csslint ignore:stop */
```

```
.e-cart-icon {
    font-family: 'cart' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-link::before {
    content: '\e700';
}
</style>
```

NAVIGATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public IActionResult Navigation()
        {
            List<object> items = new List<object>();
            items.Add(new
            {
                text = "Flipkart",
                iconCss = "e-cart-icon e-link",
                url= "https://www.google.co.in/search?q=flipkart"
            });
            items.Add(new
            {
                text = "Amazon",
                iconCss = "e-cart-icon e-link",
                url= "https://www.google.co.in/search?q=amazon"
            });
            items.Add(new
            {
                text = "Snapdeal",
                iconCss = "e-cart-icon e-link",
                url= "https://www.google.co.in/search?q=snapdeal"
            });
            ViewBag.items = items;
            return View();
        }
    }
}
```

Template and Multilevel nesting

Template

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)ViewBag.menuItems).BeforeItemRender("beforeItemRender").Render()
<script>
    function beforeItemRender(args) {
        var shortCutSpan = createElement('span');
        var text = args.item.text;
        var shortCutText = text === 'Save as...' ? 'Ctrl + S' : (text
        === 'View page source' ?
            'Ctrl + U' : 'Ctrl + Shift + I');
        shortCutSpan.textContent = shortCutText;
        args.element.appendChild(shortCutSpan);
        shortCutSpan.setAttribute('class', 'shortcut');
    }
</script>
<style>
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
</style>
```

TEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public IActionResult Template()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "Save as... "
            });
            menuItems.Add(new
            {
                text = "View page source "
            });
        }
    }
}
```

```

        menuItems.Add(new
        {
            text = "Inspect  "
        });
        ViewBag.menuItems = menuItems;
        return View();
    }
}

```

Multilevel nesting

CSHTML

```

<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((I
Enumerable<object>)ViewBag.menuItems).Render()
<style>
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
</style>

```

NESTING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public ActionResult Nesting()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "Show All Bookmarks"
            });
            menuItems.Add(new
            {
                text = "Bookmarks Toolbar",
                items = new List<object>()
                {
                    new {
                        text = "Most Visited",

```



```

        items = new List<object>()
        {
            new {
                text = "Google"
            },
            new {
                text = "Gmail"
            }
        },
        new {
            text = "Recently Added"
        }
    });
    ViewBag.menuItems = menuItems;
    return View();
}
}

```

See Also

- [Populate menu items with data source](#)

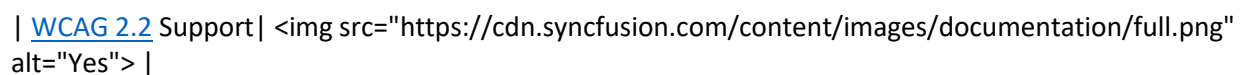
Accessibility

The Context menu component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Context menu component is outlined below.

| Accessibility Criteria | Compatibility |

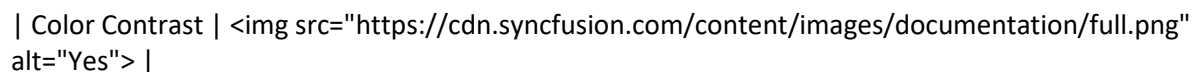
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |

| Keyboard Navigation Support |  alt="Yes" > |

```
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Context menu component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Context menu component:

Attributes	Purpose
role	Indicates Context menu component popup as menu , and the popup items as menuitem .
aria-haspopup	Indicates the availability and type of interactive popup element.
aria-expanded	Indicates whether the subtree can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed.
aria-label	Indicates the menu item text.

Keyboard interaction

The Context menu component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Context menu component.

Press	To do this
Esc	Closes the opened sub menu.
Enter	Selects the focused item.
Up	Navigates up or to the previous menu item.
Down	Navigates down or to the next menu item.

| Left | Close the current sub menu and navigates to the parent menu. |

| Right | Navigates and open the next sub menu. |

Ensuring accessibility

The Context menu component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Context menu component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Context menu component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Styles and Appearances

To modify the ContextMenu appearance, you need to override the default CSS of ContextMenu component. Find the list of CSS classes and its corresponding section in ContextMenu component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ----- | ----- |

| .e-contextmenu-wrapper | To customize the context menu wrapper |

| .e-contextmenu-wrapper .e-menu-parent | To customize the context menu items |

| .e-contextmenu-wrapper ul .e-menu-item.e-selected .e-caret::before | To customize the context menu caret icon |

| .e-contextmenu-wrapper ul .e-menu-item .e-menu-icon::before | To customize the icons of the context menu |

How To

Data Binding in Context Menu Control

The below example demonstrates how to bind local data source to the ContextMenu and separator is added using `insertAfter` method.

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)ViewBag.menuItems).BeforeItemRender("beforeItemRender").Render()
<script>
    function beforeItemRender(args) {
        if (!args.item.text) {
            args.element.classList.add('e-separator');
        }
    }
</script>
<style>
button {
```

```
margin: 20px 0 0 5px;
}
#target {
border: 1px dashed;
height: 150px;
padding: 10px;
position: relative;
text-align: justify;
color: gray;
user-select: none;
}
</style>
```

DATABIND.CS

```
public IActionResult Index()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        id=1,
        text = "View"
    });
    menuItems.Add(new
    {
        id = 2,
        text = "Sort By"
    });
    menuItems.Add(new
    {
        id = 5,
        text = ""
    });
    menuItems.Add(new
    {
        id = 3,
        text = "New"
    });
    menuItems.Add(new
    {
        id = 4,
        text = "Ascending",
        parentId = 2
    });
    ViewBag.menuItems = menuItems;
    return View();
}
```

Open and close ContextMenu

ContextMenu can be opened and closed programmatically whenever required by using the open and close methods.

In the following example, the ContextMenu is opened using the `open` method at the specified position using `top` and `left`. Also, ContextMenu is closed using `close` method on ContextMenu item click or document click.

CSHTML

```
@Html.EJS().Button("button").Content("Open ContextMenu").Render()
@Html.EJS().ContextMenu("contextmenu").Items((IEnumerable<object>)ViewBag.menuItems).Render()
<script>
    document.addEventListener("DOMContentLoaded", function () {
        document.getElementById('button').onclick = function () {
            ej.base.getComponent(document.getElementById("contextmenu"),
"contextmenu").open(60, 20);
        }
    });
</script>
<style>
    button {
        margin: 20px 0 0 5px;
    }
</style>
</style>
```

OPENCLOSE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public ActionResult OpenClose()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "Cut"
            });
            menuItems.Add(new
            {
                text = "Copy"
            });
            menuItems.Add(new
            {
                text = "Paste"
            });
            ViewBag.menuItems = menuItems;
            return View();
        }
    }
}
```

Template

Table in Sub ContextMenu

Menu items of the ContextMenu can be customized according to the requirement. The section explains about how to customize table template in sub menu item.

This can be achieved by appending table layout while **li** rendering by using [beforeItemRender](#) event.

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>) ViewBag.menuItems).BeforeItemRender("beforeItemRender").Render()
<script>
    function beforeItemRender(args) {
        if (args.item.text === 'Insert') {
            args.element.innerText = '';
            args.element.appendChild(this.createTable());
            args.element.style.height = '105px';
            args.element.classList.add('e-hover-list');
        }
        if (args.item.text === 'Insert Table') {
            args.element.classList.add('e-hover-list');
        }
    }
    function createTable() {
        var table = document.createElement('table');
        for (let i: number = 0; i < 5; i++) {
            let row: HTMLElement = document.createElement('tr');
            table.appendChild(row);
            for (let j: number = 0; j < 6; j++) {
                let col: HTMLElement = document.createElement('td');
                row.appendChild(col);
                col.setAttribute('class', 'e-data');
            }
        }
        return table;
    }
</script>
<style>
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
/* csslint ignore:start */
@@font-face {
    font-family: 'ddb-icons';
    src:
        url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAIAIAAAwAgTlMvMj0gSRkAAAEoAAAVmNtYXDnE+dkAAABlAAAADxnbHlmlh3
```

```

3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEWBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADYgABAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAEEAAAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAptAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAwGPsAJYAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAFAAEEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAAI4AwGEEASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFx06IBkNCQIHCy8bCQG9BwIJDkG9BwIJDkG9BwIJDkG9BwI
OIxo+ExckFAQMfIkWVhCMBwYlFRYkBWcMf1YwFCALDAQUIxcUPhojDiaOUR4cAQvEwWSB6gtdTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMDg0sOZsaKQ4ONZcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQA+kABQANABcAHwAAARUzFSERAYERizU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAC
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVl+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAQAAAAAAQAJAEEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACGAsADCAAQAAAAAACWASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHCAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlcn2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Zlcn2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBJAG8AbgBzAGQAZABiAC0AaQBJAG8AbgBzAFY
AZQByAHMAaQBVAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBUAGcAIABTAKhAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAQcBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxZGZvbG9uQ
OcGFyYS1tYXJrLS0tMDMAAA==) format('trueType');
    font-weight: normal;
    font-style: normal;
}
/* csslint ignore:stop */
.e-cm-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-cut::before {
    content: '\e700';
}
.e-copy::before {
    content: '\e70a';
}
.e-paste::before {
    content: '\e701';
}
.e-link::before {
    content: '\e290';
}

```

```
}  
.e-table::before {  
    content: '\e705';  
}  
</style>
```

TABLE.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
namespace WebApplication1.Controllers  
{  
    public class ContextMenuController : Controller  
    {  
        public ActionResult Table()  
        {  
            List<object> menuItems = new List<object>();  
            menuItems.Add(new  
            {  
                text = "Cut",  
                iconCss = "e-cm-icons e-cut"  
            });  
            menuItems.Add(new  
            {  
                text = "Copy",  
                iconCss = "e-icons e-copy"  
            });  
            menuItems.Add(new  
            {  
                text = "Paste",  
                iconCss = "e-cm-icons e-paste"  
            });  
            menuItems.Add(new  
            {  
                separator = true  
            });  
            menuItems.Add(new  
            {  
                text = "Link",  
                iconCss = "e-icons e-link"  
            });  
            menuItems.Add(new  
            {  
                text = "Table",  
                items = new List<object>()  
                {  
                    new {  
                        text = "Insert Table",  
                    },  
                    new {  
                        separator = true  
                    },  
                    new {  

```



```

        text = "Insert"
    }
    }
    });
    ViewBag.menuItems = menuItems;
    return View();
}
}
}

```

UI Components in ContextMenu

UI components can also be placed inside the each `li` element of ContextMenu.

In the following example, CheckBox component is placed inside each `li` element and this can be achieved by creating CheckBox component in [beforeItemRender](#) event and appending it into the `li` element.

CSHTML

```

<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)ViewBag.menuItems).BeforeClose("beforeClose").BeforeItemRender("beforeItemRender").Select("select").Render()
<script>
    function beforeClose(args) {
        args.cancel = true;
    }
    function select(args) {
        var selectedElem = args.element.parentElement.querySelectorAll('.e-selected');
        for(var i=0; i < selectedElem.length; i++){
            var ele = selectedElem[i];
            ele.classList.remove('e-selected');
        }
        var checkbox = args.element.childNodes[0] as HTMLElement;
        var frame = checkbox.querySelector('.e-frame');
        if (checkbox && frame.classList.contains('e-check')) {
            frame.classList.remove('e-check');
        } else if (checkbox) {
            frame.classList.add('e-check');
        }
    }
    var i = 1;
    function beforeItemRender(args) {
        var check = @Html.EJS().CheckBox("checkbox" + i).Label(args.element.text).Checked((args.item.text == 'Option 2') ? true : false).Render()
        args.element.innerHTML = '';
        args.element.appendChild(check);
        i++;
    }
</script>
<style>
    .list {
        display: none;
    }
</style>

```

```
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
</style>
```

UICOMPONENTS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public ActionResult UIComponents()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "Option 1"
            });
            menuItems.Add(new
            {
                text = "Option 2"
            });
            menuItems.Add(new
            {
                text = "Option 3"
            });
            ViewBag.menuItems = menuItems;
            return View();
        }
    }
}
```

Underline a character in the item text

To underline a particular character in a text, it can be handled in [beforeItemRender](#) event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((I
Enumerable<object>)ViewBag.menuItems).BeforeItemRender("beforeItemRender").R
ender()
```

```
<script>
    function beforeItemRender(args) {
        if (args.item.text === "Copy") {
            args.element.innerHTML = '<u>C</u>opy';
        }
    }
</script>
<style>
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
</style>
```

UNDERLINE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public ActionResult Underline()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "Cut"
            });
            menuItems.Add(new
            {
                text = "Copy"
            });
            menuItems.Add(new
            {
                text = "Paste"
            });
            ViewBag.menuItems = menuItems;
            return View();
        }
    }
}
```

Open a dialog on ContextMenu item click

This section explains about how to open a dialog on ContextMenu item click. This can be achieved by handling dialog open in [select](#) event of the ContextMenu.

In the following sample, Dialog will open while clicking **Save As...** item:

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((I
Enumerable<object>)ViewBag.menuItems).Select("itemSelect").Render()
@Html.EJS().Dialog("dialog").Header("Save As").Content("This file can be
saved as
PDF").Width("250px").Visible(false).Buttons(ViewBag.DialogButtons).Render()
<script>
    function dlgButtonClick() {
        ej.base.getComponent(document.getElementById('dialog'),
"dialog").hide();
    }
    function itemSelect() {
        ej.base.getComponent(document.getElementById('dialog'),
"dialog").show();
    }
</script>
<style>
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
</style>
```

DIALOGBUTTON-CORE.CS

```
public ActionResult DialogButton()
{
    ViewBag.DialogButtons = new
    {
        isPrimary = true,
        cssClass = "e-flat",
        content = "Submit",
        click = "dlgButtonClick"
    };
    return View();
}
```

Change animation settings

To change the animation of the ContextMenu, [animationSettings](#) property is used. The supported effects for ContextMenu are,

Effect	Functionality
-----	-----

- | None | Specifies the sub menu transform with no animation effect. |
- | SlideDown | Specifies the sub menu transform with slide down effect. |
- | ZoomIn | Specifies the sub menu transform with zoom in effect. |
- | FadeIn | Specifies the sub menu transform with fade in effect. |

The following sample illustrates how to open ContextMenu with **FadeIn** effect with the **duration** of **800ms**.

CSHTML

```
<div id="contextmenutarget">Right click/Touch hold to open the ContextMenu
</div>
@Html.EJS().ContextMenu("contextmenu").Target("#contextmenutarget").Items((IEnumerable<object>)ViewBag.menuItems).AnimationSettings("animation").Render()
<script>
    var animation = {
        effect: 'FadeIn',
        duration: 800
    };
</script>
<style>
#contextmenutarget {
    border: 1px dashed;
    height: 250px;
    padding: 10px;
    position: relative;
    text-align: center;
    color: gray;
    line-height: 17;
    font-size: 14px;
}
</style>
```

ANIMATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ContextMenuController : Controller
    {
        public ActionResult Animation()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "Show All Bookmarks"
            });
            menuItems.Add(new
            {
```

```
        text = "Bookmarks Toolbar",
        items = new List<object>()
        {
            new {
                text = "Most Visited",
                items = new List<object>()
                {
                    new {
                        text = "Google"
                    },
                    new {
                        text = "Gmail"
                    }
                }
            },
            new {
                text = "Recently Added"
            }
        }
    });
    ViewBag.menuItems = menuItems;
    return View();
}
}
```

Dashboard Layout

Getting Started with ASP.NET MVC Dashboard Layout Control

This section briefly explains about how to include [ASP.NET MVC Dashboard Layout](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5

NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

`

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

`

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Dashboard Layout control

Now, add the Syncfusion ASP.NET MVC Dashboard Layout control in **~/Views/Home/Index.cshtml** page by the following ways.

- Defined the panels property as the attribute in the content template.
- Using the [Panels](#) property through helper.

Setting the panels property using content template

You can render the Dashboard Layout control by adding the panels property as the attribute to the content template. Add the content template with panel definition for Dashboard Layout into your `index.cshtml` page which is present under `Views/Home` folder.

In the following sample, the dashboard layout is rendered with [Panels](#) property using content template.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- DashboardLayout element declaration -->

@Html.EJS().DashboardLayout("defaultLayout").Columns(6).CellSpacing(Model.cellSpacing).ContentTemplate(@<div>
    <div id="one" class="e-panel" data-row="0" data-col="0" data-sizeX="1" data-sizeY="1">
        <div class="e-panel-container">
            <div class="text-align">0</div>
        </div>
    </div>
    <div id="two" class="e-panel" data-row="1" data-col="0" data-sizeX="1" data-sizeY="2">
        <div class="e-panel-container">
            <div class="text-align">1</div>
        </div>
    </div>
    <div id="three" class="e-panel" data-row="0" data-col="1" data-sizeX="2" data-sizeY="2">
        <div class="e-panel-container">
            <div class="text-align">2</div>
        </div>
    </div>
    <div id="four" class="e-panel" data-row="2" data-col="1" data-sizeX="1" data-sizeY="1">
        <div class="e-panel-container">
            <div class="text-align">3</div>
        </div>
    </div>
    <div id="five" class="e-panel" data-row="2" data-col="2" data-sizeX="2" data-sizeY="1">
        <div class="e-panel-container">
            <div class="text-align">4</div>
        </div>
    </div>
    <div id="six" class="e-panel" data-row="0" data-col="3" data-sizeX="1" data-sizeY="1">
        <div class="e-panel-container">
            <div class="text-align">5</div>
        </div>
    </div>
</div>
```



```

        <div id="seven" class="e-panel" data-row="1" data-col="3" data-
sizeX="1" data-sizeY="1">
            <div class="e-panel-container">
                <div class="text-align">6</div>
            </div>
        <div id="eight" class="e-panel" data-row="0" data-col="4" data-
sizeX="1" data-sizeY="3">
            <div class="e-panel-container">
                <div class="text-align">7</div>
            </div>
        </div> .Render()
    <!-- end of dashboardlayout element -->
</div>
</div>
<style>
/* DashboardLayout element styles */
#defaultLayout {
    padding: 10px;
}
#defaultLayout .e-panel .e-panel-container {
    vertical-align: middle;
    font-weight: 600;
    font-size: 20px;
    text-align: center;
}
.text-align {
    line-height: 160px;
}
</style>

```

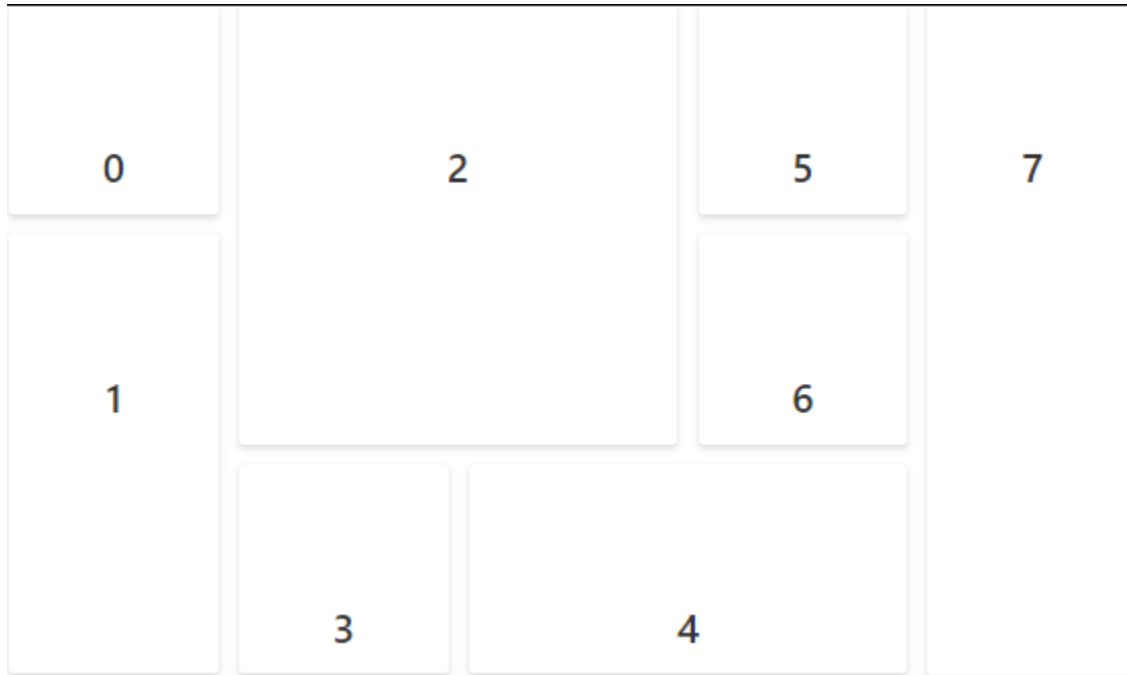
HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Dashboard Layout control will be rendered in the default web browser.



Setting the panels property using helper

You can render the Dashboard Layout control by using the **panels** property through helper.

In the following sample, the dashboard layout is rendered with [Panels](#) property using tag helper.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- DashboardLayout element declaration -->

        @Html.EJS().DashboardLayout("defaultLayout").Columns(6).CellSpacing(Model.cellSpacing).Panels(Panel =>
            {
                Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div>0</div>").Add();
                Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div>1</div>").Add();
                Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div>2</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div>3</div>").Add();
                Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div>4</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div>5</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div>6</div>").Add();
            }).Render()
        <!-- end of dashboardlayout element -->
    </div>
</div>
<style>
    /* DashboardLayout element styles */
    #defaultLayout {
        padding: 10px;
    }
</style>
```

```
#defaultLayout .e-panel .e-panel-container {  
    vertical-align: middle;  
    font-weight: 600;  
    font-size: 20px;  
    text-align: center;  
    line-height: 160px;  
}  
</style>
```

HOMECONTROLLER.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using Microsoft.AspNetCore.Mvc;  
namespace WebApplication.Controllers  
{  
    public class HomeController : Controller  
    {  
        public class spacingModel  
        {  
            public double[] cellSpacing { get; set; }  
        }  
        public ActionResult Index()  
        {  
            spacingModel modelValue = new spacingModel();  
            modelValue.cellSpacing = new double[] { 10, 10 };  
            return View(modelValue);  
        }  
    }  
}
```



Note: [View Sample in GitHub.](#)

Configuring the layout

The entire layout dimensions are assigned based on the height and width of the parent element. Hence, a responsive or static layout can be created by assigning a percentage or static dimension values to the parent element. The layout adapts to mobile resolutions by transforming the entire layout into a stacked orientation, so that, the panels will be displayed in a vertical column.

The **Dashboard Layout** is a grid structured component which can be split into subsections of equal size known as cells. The total number of cells in each row is defined by using the [columns](#) property of the component. The width of each cell will be auto calculated based on the total number of cells placed in a row and the height of a cell will be same as that of its width. However, the height of these cells can also be configured to any desired size using the [cellAspectRatio](#) property (cellwidth/cellheight ratio) which defines the cell width to height ratio.

The number of rows within the layout has no limits and can have any number of rows based on the panels count and position. Panels which acts as data containers will be placed or positioned over these cells.

Modifying cell size

In a dashboard, the data to be held by the panel in a cell may be of different size, hence different cell dimensions may be required in different scenarios. In this case, the size of these grid cells can be modified to the required size using the [columns](#) and [cellAspectRatio](#) properties.

The following sample demonstrates how to modify a cell size using the [columns](#) and [cellAspectRatio](#) properties. In the following sample, the width of the parent element is divided into 5 equal cells based on the columns property value resulting the width of each cell as 100px. The height of these cells will be 50px based on the cellAspectRatio value 100/50 (i.e., for every 100px of width, 50px will be the height of the cell).

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- DashboardLayout element declaration -->

@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model
.cellSpacing).CellAspectRatio(Model.aspectRatio).Panels(Panel =>
    {
        Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div
class='content'>0</div>").Add();
        Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div
class='content'>1</div>").Add();
        Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div
class='content'>2</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div
class='content'>3</div>").Add();
        Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div
class='content'>4</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div
class='content'>5</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div
class='content'>6</div>").Add();
    }).Render()
```

```

        <!-- end of dashboardlayout element -->
    </div>
</div>
<style>
    /* DashboardLayout element styles */
    #dashboard_layout .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 500;
        font-size: 20px;
        text-align: center;
        line-height: 45px;
    }
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
            public double aspectRatio { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            modelValue.aspectRatio = 100 / 50;
            return View(modelValue);
        }
    }
}

```



Setting cell spacing

The spacing between each panel in a row and column can be defined using the [cellSpacing](#) property. Adding spacing between the panels will make the layout effective and provides a clear data representation.

The following sample demonstrates the usage of the [cellSpacing](#) property, which helps in a neat and clear representation of a data.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- Dashboardlayout element declaration -->

        @Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model
            .cellSpacing).Panels(Panel =>
            {

                Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div>0</div>").Add();
                Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div>1</div>").Add();
                Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div>2</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div>3</div>").Add();
                Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div>4</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div>5</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div>6</div>").Add();
            }).Render()
        </div>
    </div>
    <!-- end of dashboardlayout element -->
    <style>
        /* DashboardLayout element styles */
        #dashboard_layout .e-panel .e-panel-container {
            vertical-align: middle;
            font-weight: 600;
            font-size: 20px;
            text-align: center;
            line-height: 90px;
        }
    </style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
```

```

public class HomeController : Controller
{
    public class spacingModel
    {
        public double[] cellSpacing { get; set; }
    }
    public ActionResult Index()
    {
        spacingModel modelValue = new spacingModel();
        modelValue.cellSpacing = new double[] { 20, 20 };
        return View(modelValue);
    }
}

```



Graphical representation of the layout

These cells combinedly forms a grid-structured layout which will be hidden initially. This grid structured layout can be made visible by enabling the [showGridLines](#) property, which clearly pictures the cells split-up within the layout. These gridlines will be helpful in panels sizing and placement within the layout during initial designing of a dashboard.

In the following sample, the gridlines indicate the cells split-up of the layout and the data containers placed over these cells are known as panels.

CSHTML

```

@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- DashboardLayout element declaration -->
    
```

```

@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).ShowGridLines(true)
.CellSpacing(Model.cellSpacing).Panels(Panel =>
{
    Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div>1</div>").Add();
    Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div>2</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div>3</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div>4</div>").Add();
}).Render()
<!-- end of dashboardlayout element -->
</div>
</div>
<style>
/* DashboardLayout element styles */
#dashboard_layout .e-panel .e-panel-container {
    vertical-align: middle;
    font-weight: 600;
    font-size: 20px;
    text-align: center;
    line-height: 90px;
}
</style>

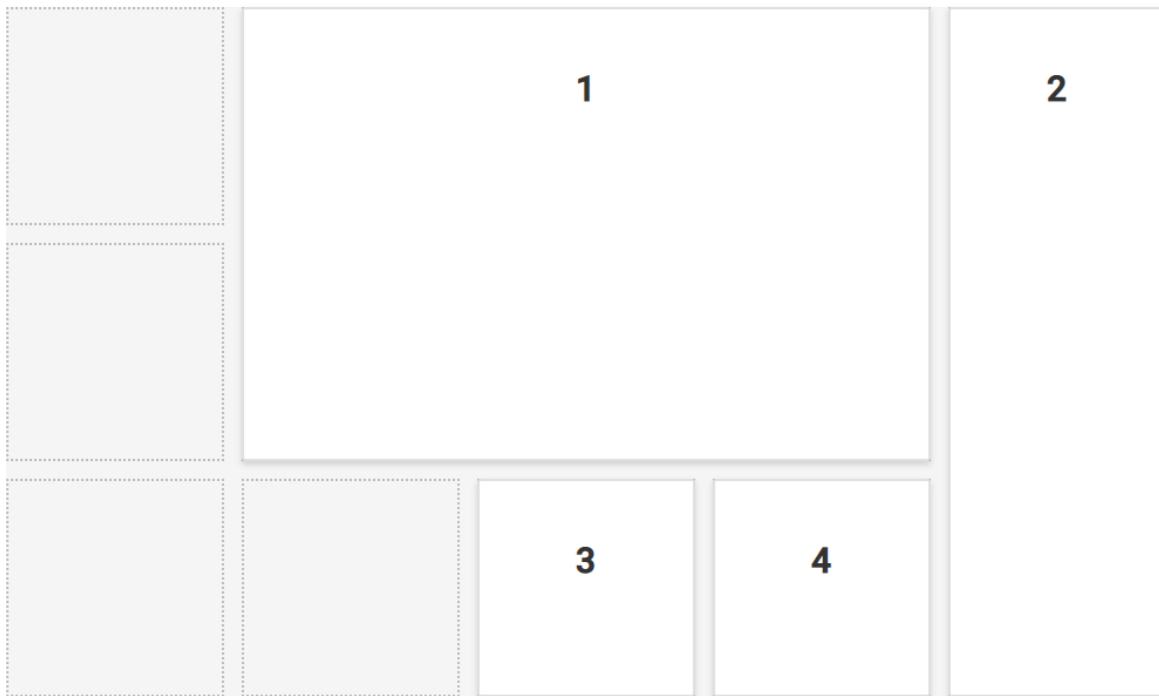
```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}

```

Rendering component in right-to-left direction

It is possible to render the Dashboard Layout in right-to-left direction by setting the [enableRtl](#) API to true.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>

@Html.EJS().DashboardLayout("dashboard_default").Columns(5).CellSpacing(Model
1.cellSpacing).EnableRtl(true).Panels(Panel =>
    {

Panel.Id("Panel0").SizeX(1).SizeY(1).Row(0).Col(0).Header("<div>Panel
0</div>").Content("<div class='content'></div>").Add();

Panel.Id("Panel1").SizeX(3).SizeY(2).Row(0).Col(1).Header("<div>Panel
1</div>").Content("<div class='content'></div>").Add();

Panel.Id("Panel2").SizeX(1).SizeY(3).Row(0).Col(4).Header("<div>Panel
2</div>").Content("<div class='content'></div>").Add();

Panel.Id("Panel3").SizeX(1).SizeY(1).Row(1).Col(0).Header("<div>Panel
3</div>").Content("<div class='content'></div>").Add();

Panel.Id("Panel4").SizeX(2).SizeY(1).Row(2).Col(1).Header("<div>Panel
4</div>").Content("<div class='content'></div>").Add();

Panel.Id("Panel5").SizeX(1).SizeY(1).Row(2).Col(2).Header("<div>Panel
5</div>").Content("<div class='content'></div>").Add();
```

```

Panel.Id("Panel6").SizeX(1).SizeY(1).Row(2).Col(3).Header("<div>Panel
6</div>").Content("<div class='content></div>").Add();
    }).Render()
</div>
</div>
<style>
    #dashboard_default .e-panel .e-panel-container .content {
        font-weight: 600;
        text-align: center;
        margin-top: 10px;
        line-height: 100%;
    }
    .e-dashboard-layout.e-control .e-panel .e-panel-container .e-panel-
header {
        color: rgba(0, 0, 0, 0.61);
    }
    .e-panel .e-header-text {
        padding: 12px 0 12px 0;
    }
    #dashboard .e-panel-container .e-panel-header {
        border-bottom: 1px solid #888383;
    }
    .e-dashboard-layout.e-control .e-panel:hover {
        border: 0px;
    }
    .e-dashboard-layout.e-control .e-panel .e-panel-header {
        font-size: 15px;
        font-weight: 500;
        height: 37px;
        padding: 10px;
        vertical-align: middle;
        /* text-align: left; */
        border-bottom: 0.5px solid rgba(0, 0, 0, .125);
    }
    .e-panel-header {
        padding: 10px;
        margin-bottom: 0;
        background-color: rgba(0, 0, 0, .03);
    }
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
    }
}

```

```
    }
    public ActionResult Index()
    {
        spacingModel modelValue = new spacingModel();
        modelValue.cellSpacing = new double[] { 10, 10 };
        return View(modelValue);
    }
}
```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Panels

Panels are the basic building blocks of the dashboard layout component. They act as a container for the data to be visualized or presented. These panels can be positioned or resized for effective presentation of the data.

The below table represents all the available panel properties and the corresponding functionalities

PanelObject	Description
---	---
id	Specifies the id value of the panel.
row	Specifies the row value in which the panel is to be placed.
col	Specifies the column value in which the panel is to be placed.

- | **sizeX** | Specifies the width of the panel in cells count. |
- | **sizeY** | Specifies the height of the panel in cells count. |
- | **minSizeX** | Specifies the minimum width of the panel in cells count. |
- | **minSizeY** | Specifies the minimum height of the panel in cells count. |
- | **maxSizeX** | Specifies the maximum width of the panel in cells count. |
- | **maxSizeY** | Specifies the maximum height of the panel in cells count. |
- | **header** | Specifies the header template of the panel. |
- | **content** | Specifies the content template of the panel. |
- | **cssClass** | Specifies the CSS class name that can be appended with each panel element. |

Positioning of panels

The panels within the layout can be easily positioned or ordered using the [row](#) and [col](#) properties of the panels. Positioning of panels will be beneficial to represent the data in any desired order.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- DashboardLayout element declaration -->

@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model
.cellSpacing).Panels(Panel =>
    {
        Panel.Row(0).Col(0).Content("<div
class='content'>1</div>").Add();
        Panel.Row(0).Col(1).Content("<div
class='content'>2</div>").Add();
        Panel.Row(0).Col(2).Content("<div
class='content'>3</div>").Add();
        Panel.Row(1).Col(0).Content("<div
class='content'>4</div>").Add();
        Panel.Row(1).Col(1).Content("<div
class='content'>5</div>").Add();
        Panel.Row(1).Col(2).Content("<div
class='content'>6</div>").Add();
    }).Render()
    <!-- end of dashboardlayout element -->
    </div>
</div>
<style>
    /* DashboardLayout element styles */
    #dashboard_layout .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 20, 20 };
            return View(modelValue);
        }
    }
}
```

1**2****3****4****5****6**

Sizing of panels

A panel's size can be varied easily by defining the [sizeX](#) and [sizeY](#) properties. The [sizeX](#) property defines the width and the [sizeY](#) property defines height of a panel in cells count. These properties are helpful in designing a dashboard, where the content of each panel may vary in size.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- DashboardLayout element declaration -->

@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model
.cellSpacing).Panels(Panel =>
{
    Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div
class='content'>0</div>").Add();
    Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div
class='content'>1</div>").Add();
    Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div
class='content'>2</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div
class='content'>3</div>").Add();
    Panel.SizeX(2).SizeY(1).Row(2).Col(1).Content("<div
class='content'>4</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div
class='content'>5</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div
class='content'>6</div>").Add();
}).Render()
    <!-- end of dashboardlayout element -->
</div>
</div>
<style>
    /* DashboardLayout element styles */
    #dashboard_layout .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>
```

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
```

```

    {
        public double[] cellSpacing { get; set; }
    }
    public ActionResult Index()
    {
        spacingModel modelValue = new spacingModel();
        modelValue.cellSpacing = new double[] { 20, 20 };
        return View(modelValue);
    }
}

```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Header and content of panels

The dashboard layout component is mostly used to represent the data used for monitoring or managing a process. These data or any HTML template can be placed as the [content](#) of a panel using the content property. Also, word or phrase that summarize the panel's content can be added as the header on the top of each panel using the [header](#) property of the panel.

CSHTML

```

@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- Dashboardlayout element declaration -->
    </div>
</div>

```

```

@Html.EJS().DashboardLayout("dashboard_default").Columns(6).CellSpacing(Mode
1.cellSpacing).Panels(Panel =>
{
    Panel.Id("Panel0").SizeX(1).SizeY(1).Row(0).Col(0).Header("<div>Panel
0</div>").Content("<div class='content'>Panel Content</div>").Add();

    Panel.Id("Panel1").SizeX(3).SizeY(2).Row(0).Col(1).Header("<div>Panel
1</div>").Content("<div class='content'>Panel Content</div>").Add();

    Panel.Id("Panel2").SizeX(1).SizeY(3).Row(0).Col(4).Header("<div>Panel
2</div>").Content("<div class='content'>Panel Content</div>").Add();

    Panel.Id("Panel3").SizeX(1).SizeY(1).Row(1).Col(0).Header("<div>Panel
3</div>").Content("<div class='content'>Panel Content</div>").Add();

    Panel.Id("Panel4").SizeX(2).SizeY(1).Row(2).Col(1).Header("<div>Panel
4</div>").Content("<div class='content'>Panel Content</div>").Add();

    Panel.Id("Panel5").SizeX(1).SizeY(1).Row(2).Col(2).Header("<div>Panel
5</div>").Content("<div class='content'>Panel Content</div>").Add();

    Panel.Id("Panel6").SizeX(1).SizeY(1).Row(2).Col(3).Header("<div>Panel
6</div>").Content("<div class='content'>Panel Content</div>").Add();
}).Render()
</div>
</div>
<!-- end of dashboardlayout element -->
<style>
/* DashboardLayout element styles */
#dashboard_default .e-panel .e-panel-container .content {
    font-weight: 600;
    text-align: center;
    margin-top: 10px;
    line-height: 100%;
}
.e-dashboard-layout.e-control .e-panel .e-panel-container .e-panel-
header {
    color: rgba(0, 0, 0, 0.61);
}
.e-panel .e-header-text {
    padding: 12px 0 12px 0;
}
#dashboard .e-panel-container .e-panel-header {
    border-bottom: 1px solid #888383;
}
.e-dashboard-layout.e-control .e-panel:hover {
    border: 0px;
}
.e-dashboard-layout.e-control .e-panel .e-panel-header {
    font-size: 15px;
    font-weight: 500;
    height: 37px;
    padding: 10px;
    vertical-align: middle;
    /* text-align: left; */

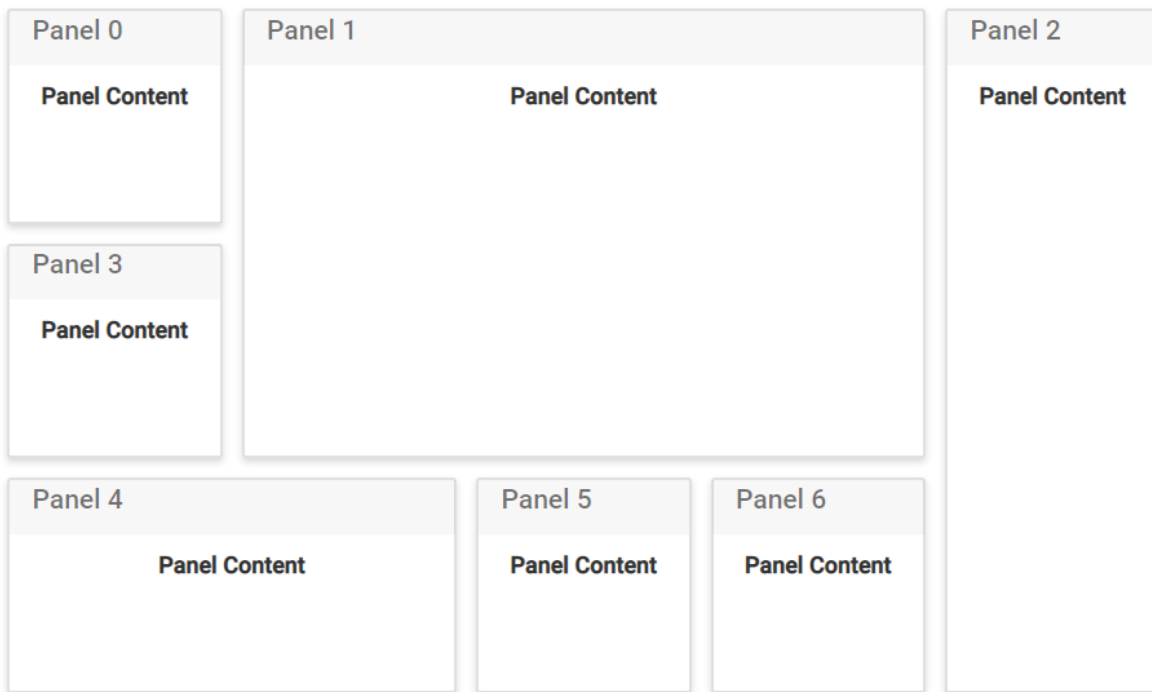
```



```
        border-bottom: 0.5px solid rgba(0, 0, 0, .125);
    }
    .e-panel-header {
        padding: 10px;
        margin-bottom: 0;
        background-color: rgba(0, 0, 0, .03);
    }
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}
```



Placing components as content

In a dashboard, components like charts, grids, maps, gauges, and more can be used to present complex data. Such components can be placed as the panel content by assigning the corresponding component element as the **content template** of the panel.

Note: You must assign the empty div element inside the content template to add the component as content and also define the `.e-panel`, `.e-panel-container`, `.e-panel-header`, and `.e-panel-content` classes while rendering the `DashboardLayout` component using content template.

The following sample demonstrates how to add ej2-chart components as the [content](#) for each panel in the `DashboardLayout` component.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div>
    <!-- DashboardLayout element declaration -->

    @Html.EJS().DashboardLayout("dashboard_default").Columns(6).CellSpacing(Model
    1.cellSpacing).DraggableHandle(".e-panel-header").Panels(Panel =>
    {
        Panel.Id("Panel1").SizeX(3).SizeY(2).Row(0).Col(3).ContentTemplate(
            @<div>
                <div class="e-panel-container">
                    <div class="e-panel-header">
                        <div>
                            Last year sales comparison
                        </div>
                    </div>
                    <div class="e-panel-content">
```

```

        <div id="column">
            <!-- Column Chart element declaration -->
            @Html.EJS().Chart("columnChart").Series(series
=>
                {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
                .XName("month")
                .YName("sales")
                .DataSource(ViewBag.chartSource).Add();
                }) .PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Height("162px").Render()

                <!-- end of column chart element -->
            </div>
        </div>
    </div>
) .Add();
Panel.Id("Panel2").SizeX(3).SizeY(2).Row(0).Col(3).ContentTemplate(
    @<div>
        <div class="e-panel-container">
            <div class="e-panel-header">
                <div>
                    Mobile browsers usage
                </div>
            </div>
            <div class="e-panel-content">
                <div id="pie1">
                    <!-- Pie Chart element declaration -->
                    @Html.EJS().AccumulationChart("pieChart1").Series(series =>
                        {
                            series.DataLabel(dl =>
                                dl.Visible(true).Name("text").Font(ft =>
                                    ft.FontWeight("600")).Position(Syncfusion.EJ2.Charts.AccumulationLabelPosition.Inside))
                                .XName("x")
                                .YName("y")
                                .Name("Browser")
                                .Radius("70%")
                                .DataSource(ViewBag.pieSource1).Add();
                            })
                                .EnableAnimation(false)
                                .EnableSmartLabels(true)
                                .Height("162px")
                                .LegendSettings(ls => ls.Visible(false))
                                .Tooltip(tp =>
                                    tp.Enable(true).Format("${point.x} : <b>${point.y}%</b>").Render()
                                )
                                <!-- end of pie chart element -->
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
) .Add();
Panel.Id("Panel3").SizeX(3).SizeY(2).Row(1).Col(0).ContentTemplate(

```

```
@<div>
    <div class="e-panel-container">
        <div class="e-panel-header">
            <div>
                Sales increase percentage
            </div>
        </div>
        <div class="e-panel-content">
            <div id="line">
                <!-- Line Chart element declaration -->
                @Html.EJS().Chart("lineChart").Series(series
=>
                    {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
                        .XName("x")
                        .YName("y")
                        .DataSource(ViewBag.lineSource).Add();
                    }).Height("162px").Render()
                <!-- end of line chart element -->
            </div>
        </div>
    </div>
</div>
).Add();
}).Render()
<!-- end of dashboardlayout element -->
</div>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        var lineObj = document.getElementById('lineChart').ej2_instances[0];
        var pieObj = document.getElementById('pieChart1').ej2_instances[0];
        var splineObj =
document.getElementById('columnChart').ej2_instances[0];
        lineObj.refresh(); pieObj.refresh(); splineObj.refresh();
    });
</script>
<style>
/* DashboardLayout element styles */
.e-dashboard-layout.e-control .e-panel .e-panel-container .e-panel-
header {
    color: rgba(0, 0, 0, 0.61);
}
.e-panel .e-header-text {
    padding: 12px 0 12px 0;
}
#dashboard .e-panel-container .e-panel-header {
    border-bottom: 1px solid #888383;
}
.e-dashboadrd-layout.e-control .e-panel:hover {
    border: 0px;
}
.e-dashboadrd-layout.e-control .e-panel .e-panel-header {
    font-size: 15px;
    font-weight: 500;
    height: 37px;
    padding: 10px;
```

```
vertical-align: middle;
/* text-align: left; */
border-bottom: 0.5px solid rgba(0, 0, 0, .125);
}
.e-panel-header {
padding: 10px;
margin-bottom: 0;
background-color: rgba(0, 0, 0, .03);
}
.e-dashboard-layout.e-control .e-panel .e-panel-header {
height: 30px
}
.e-dashboard-layout.e-control .e-panel .e-panel-header div {
text-align: center;
}
.header {
padding: 5px;
display: inline-block;
}
.e-panel-content {
height: 162px;
}
/* custom generated icons styles */
@@font-face {
font-family: 'e-icons';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQYkCgIAAAIkAAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAACAACAAAAAAAAAAAAEAAAAQAIAAwAAAAA
AAgAAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAgAAAAAAMAAAAUAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAQACAAYABAADAAGABWAKAAKABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoDl1vwq6I0B64wB640AAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPah8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHB0ZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBWgICaKJCgokCwsLDAwMGRoD4gMEBwQFBQYGBWgICaKKGs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTEUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjMlKwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAWIBAQIDAwmEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAgMDAwQE0AUEAWMDAgEBAgMDAwQ
FfaUEAWMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAGIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEBAEACAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAWMDAgEBAgMDAwQFaAQEAwQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxyVExEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHBoZFxyVExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAEAOAQcuASc+ATceAQEHBgcNjgYPAQYWHwE
GFBChDgEfAR4BPWEWHwEeATsBMjY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYmhhSBhIrgg
```

```

FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4za+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEAAgQEBAAA1Dw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUFHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFBQTExITEREREBAPDw00DAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA
QERERERUUFBQTExITEREREBAPDw00DAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwvNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFQRUEXIREQ8PDg0MCwoJCACFBQMCAGMFBQCICQoLDA00Dw8
RERITFBQVfYhYWGBCYGRkZGhoZGRkYFhgWfYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQCHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBAAPDw8ODg4NDQIwCgkICQCHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehM
TFBQUFRoZGRkYFhgWfYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgMDQ4PDxEREhMUFBU
WFhYFhgZGRkaGhkZGRgXGBYWFhUUFBMSErPDw4NDAsKCQgHBQUdAgIDBQUHCAkKCwvNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMHw8RDw8hLw8RPw6aAswNDgwMDAsKCgIBwU
FAwIBAQIDBQUHCAgKCgMDAwODf00DQ4MDAwLCgoICAcFBQMCAGQECawUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgSLDA0NDv00Dg0NDAsLCgoICAcFBQMCAGQECawUFBwgICgoLCwvNDQ4CzA4NDQwLCwo
KCAGHBQUdAgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAA0AAQABAAAAAACAACADgABAAAAA
DAA0AFQABAAAAAEEAA0AIgABAAAAAFAASALwABAAAAAAGAA0AOgABAAAAAAKACwArWABAAA
AAAALABIAcWADAAEECQAAAAIAhQADAAEECQABABoAhWADAAEECQACAA4AoQADAAEECQADABoArWA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEWADAAEECQALACQ
BayB1LW1jb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm91LW1jb25zLW1ldHJvVmVyc2lvbiA
xLjB1LW1jb25zLW1ldHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBJAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbWBlAC0AaQBJAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBVAG4AIAAxAC4AMABlAC0AaQBJAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgBlAHMAaQBVAG4AIAAB
NAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgBlAHMAaQBVAG4ALgBjAG8
AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEQAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQRlc2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHMY2hldnJvbi1sZWZ0CE1UX0NsZWfyDE1UX0plbmtdtYwlsCwR
zdG9wAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.handler.burg-icon:before {
    content: '\e10d';
    font-size: 16px;
}
.handler.burg-icon {
    float: right;
    padding-top: 2%;
}
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel

```

```
{
    public double[] cellSpacing { get; set; }
}
public class ChartData
{
    public string month;
    public double sales;
}
public class LineData
{
    public double x;
    public double y;
}
public class PieData
{
    public string x;
    public double y;
    public string text;
}
public ActionResult Index()
{
    List<ChartData> chartData = new List<ChartData>
    {
        new ChartData {
            month = "Jan",
            sales = 35,
        },
        new ChartData {
            month = "Feb",
            sales = 28,
        },
        new ChartData {
            month = "Mar",
            sales = 34,
        },
        new ChartData {
            month = "Apr",
            sales = 32,
        },
        new ChartData {
            month = "May",
            sales = 40,
        },
        new ChartData {
            month = "Jun",
            sales = 32,
        },
        new ChartData {
            month = "Jul",
            sales = 35,
        },
        new ChartData {
            month = "Aug",
            sales = 55,
        },
        new ChartData {
            month = "Sep",
```

```

        sales = 38,
    },
    new ChartData {
        month = "Oct",
        sales = 30,
    },
    new ChartData {
        month = "Nov",
        sales = 25,
    },
    new ChartData {
        month = "Dec",
        sales = 32,
    }
};
ViewBag.chartSource = chartData;
List<LineData> lineData = new List<LineData>
{
    new LineData {
        x = 2013,
        y = 28,
    },
    new LineData {
        x = 2014,
        y = 25,
    },
    new LineData {
        x = 2015,
        y = 26,
    },
    new LineData {
        x = 2016,
        y = 27,
    },
    new LineData {
        x = 2017,
        y = 32,
    },
    new LineData {
        x = 2018,
        y = 35,
    }
};
ViewBag.lineSource = lineData;
List<PieData> pieData1 = new List<PieData>
{
    new PieData {
        x = "Chrome",
        y = 37,
        text = "37%",
    },
    new PieData {
        x = "UC Browser",
        y = 17,
        text = "17%",
    },
    new PieData {

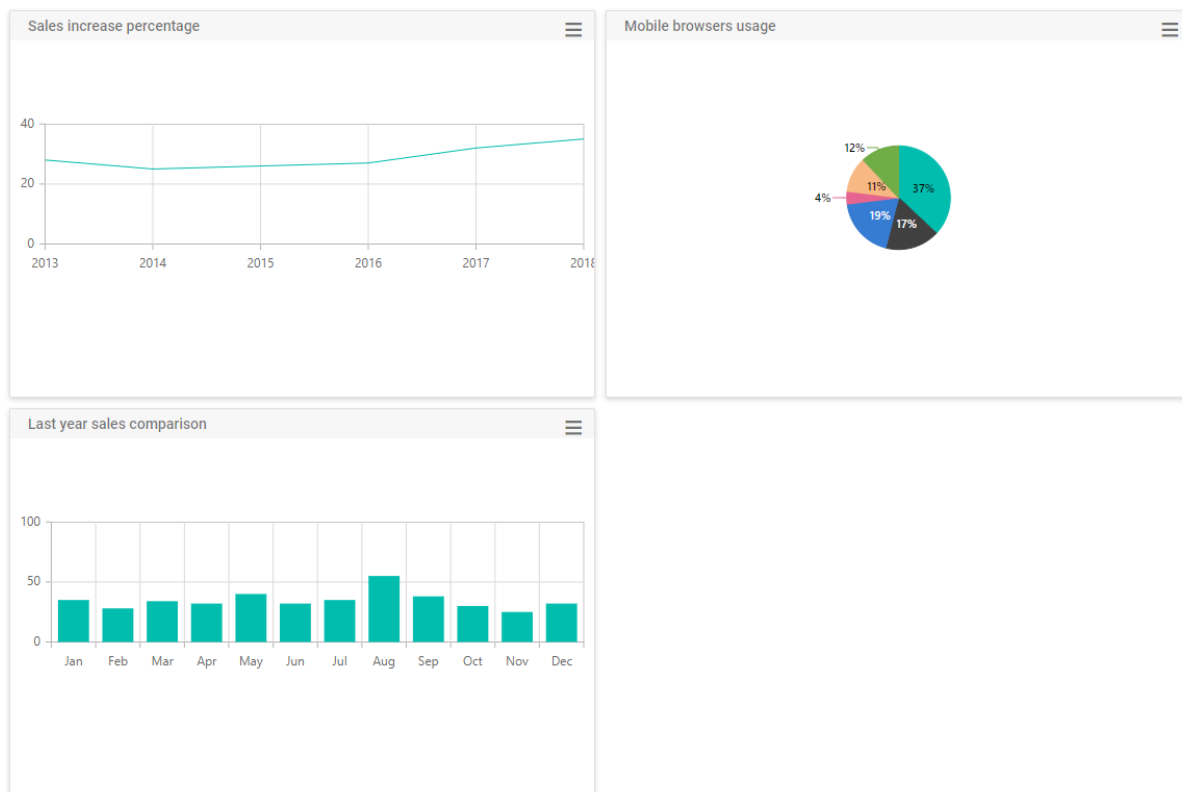
```



```

        x = "iPhone",
        y = 19,
        text = "19%",
    },
    new PieData {
        x = "Others",
        y = 4,
        text = "4%",
    },
    new PieData {
        x = "Opera",
        y = 11,
        text = "11%",
    },
    new PieData {
        x = "Android",
        y = 12,
        text = "12%",
    }
    };
    ViewBag.pieSource1 = pieData1;
    spacingModel modelValue = new spacingModel();
    modelValue.cellSpacing = new double[] { 10, 10 };
    return View(modelValue);
}
}
}

```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Adding and removing panels dynamically

In real-time cases, the data being presented within the dashboard should be updated frequently which includes adding or removing the data dynamically within the dashboard. This can be easily achieved by using the `addPanel` and `removePanel` public methods of the component.

Add or remove panels dynamically

Panels can be added dynamically by using the `addPanel` public method by passing the panel property as parameter. Also, they can be removed dynamically by using the `removePanel` public method by passing the panel id value as a parameter.

It is also possible to remove all the panels in a Dashboard Layout by calling `removeAll` method.

```
`js
```

```
dashboard.removeAll();
```

The following sample demonstrates how to add and remove the panels dynamically in the dashboard layout component. Here, panels can be added in any desired position of required size by selecting them in the numeric boxes and clicking add button and remove them by selecting the ID of the panel.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div id="api_property">
    <div style="display:inline-block">
        <div style="padding:5px;padding-right: 40px;">
            <!-- NumericTextBox element declaration -->

@Html.EJS().NumericTextBox("sizeX").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Never).Placeholder("Size X").Min(1).Max(5).Width("150px").Render()
        </div>
        <div style="padding:5px;">
            <!-- NumericTextBox element declaration -->

@Html.EJS().NumericTextBox("sizeY").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Never).Placeholder("Size Y").Min(1).Max(5).Width("150px").Render()
        </div>
    </div>
    <div style="display:inline-block">
        <div style="padding:5px;padding-right: 40px;">
            <!-- NumericTextBox element declaration -->

@Html.EJS().NumericTextBox("row").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Never).Placeholder("Row").Min(0).Max(5).Width("150px").Render()
        </div>
        <div style="padding:5px;">
            <!-- NumericTextBox element declaration -->
```

```

@Html.EJS().NumericTextBox("column").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Never).Placeholder("Column").Min(0).Max(4).Width("150px").Render()
</div>
</div>
<div style="padding:5px; width: 240px;">
    <!-- Dropdownlist element declaration -->
    @Html.EJS().DropDownList("dropdown").Placeholder("Select a id value").DataSource(ViewBag.data).Width("150px").Index(0).Render()
</div>
<div style="padding:5px;">
    <div style="width:100%">
        <!-- Button element declaration -->
        @Html.EJS().Button("add").CssClass("e-primary").Content("Add Panel").Render()
        @Html.EJS().Button("remove").CssClass("e-danger").Content("Remove Panel").Render()
    </div>
</div>
</div>
<div class="control-section" style="padding-top: 15px;" id="control_dash">
    <div>
        <!-- Dashboardlayout element declaration -->

@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model.cellSpacing).Panels(Panel =>
    {
        Panel.Id("Panel0").SizeX(1).SizeY(1).Row(0).Col(0).Content("<div class='content'>0</div>").Add();
        Panel.Id("Panel1").SizeX(3).SizeY(2).Row(0).Col(1).Content("<div class='content'>1</div>").Add();
        Panel.Id("Panel2").SizeX(1).SizeY(3).Row(0).Col(4).Content("<div class='content'>2</div>").Add();
        Panel.Id("Panel3").SizeX(1).SizeY(1).Row(1).Col(0).Content("<div class='content'>3</div>").Add();
        Panel.Id("Panel4").SizeX(2).SizeY(1).Row(2).Col(0).Content("<div class='content'>4</div>").Add();
        Panel.Id("Panel5").SizeX(1).SizeY(1).Row(2).Col(2).Content("<div class='content'>5</div>").Add();
        Panel.Id("Panel6").SizeX(1).SizeY(1).Row(2).Col(3).Content("<div class='content'>6</div>").Add();
    }).Render()
    </div>
</div>
<!-- end of dashboardlayout element -->
<script>
    document.addEventListener('DOMContentLoaded', function () {
        // Create a instances for dashboard layout
        var dashboardObject =
document.getElementById('dashboard_layout').ej2_instances[0];
        // Remove panel in dashboard layout
        document.getElementById('remove').onclick = function (e) {
            var dropdownObject =
document.getElementById('dropdown').ej2_instances[0];
            dashboardObject.removePanel(dropdownObject.value);

```

```

dropdownObject.dataSource.splice(dropdownObject.dataSource.indexOf(dropdownO
bject.value), 1);
    dropdownObject.value = null;
    dropdownObject.refresh();
};
var count = 7;
// Add panel in dashboard layout
document.getElementById('add').onclick = function (e) {
    var sizeX = document.getElementById('sizeX').ej2_instances[0];
    var sizeY = document.getElementById('sizeY').ej2_instances[0];
    var row = document.getElementById('row').ej2_instances[0];
    var column = document.getElementById('column').ej2_instances[0];
    var dropdownObject =
document.getElementById('dropdown').ej2_instances[0];
    var panel = [{
        id: count.toString() + "Panel", sizeX: parseInt(sizeX.value,
10), sizeY: parseInt(sizeY.value, 10), row: parseInt(row.value, 10), col:
parseInt(column.value, 10),
        content: "<div class='content'>" + count + "</div>"
    }];
    dropdownObject.dataSource.push("Panel" + count.toString());
    dropdownObject.refresh();
    count = count + 1;
    dashboardObject.addPanel(panel[0]);
};
});
</script>
<style>
/* DashboardLayout element styles */
#dashboard_layout .e-panel .e-panel-container .content {
    vertical-align: middle;
    font-weight: 500;
    font-size: 20px;
    text-align: center;
    line-height: 45px;
}
/* sample level styles */
td {
    padding: 5px;
}
#control_dash {
    display: block;
    width: 60%;
    float: left;
}
#api_property {
    display: inline-block;
    margin: 30px;
}
</style>

```

HOMECONTROLLER.CS

```

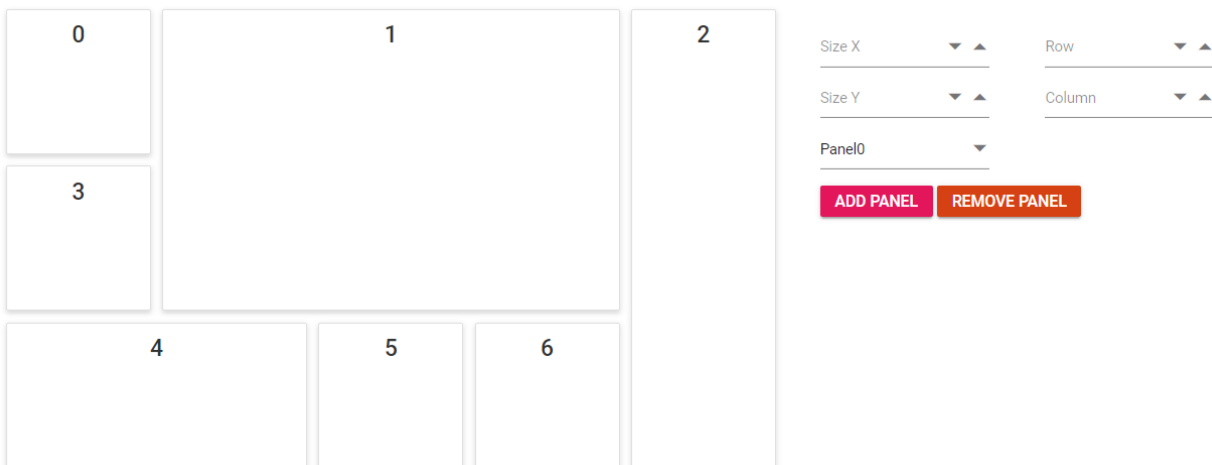
using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            ViewBag.data = new string[] { "Panel0", "Panel1", "Panel2",
            "Panel3", "Panel4", "Panel5", "Panel6" };
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}

```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Dragging of panels

The Dashboard Layout component is provided with dragging functionality to drag and reorder the panels within the layout. While dragging a panel, a holder will be highlighted below the panel indicating the panel placement on panel drop. This helps the user to decide whether to place the panel in the current position or revert to previous position without disturbing the layout.

If one or more panels collide while dragging, then the colliding panels will be pushed towards the left or right or top or bottom direction where an adaptive space for the collided panel is available. The position changes of these collided panels will be updated dynamically during dragging of a panel, so the user can conclude whether to place the panel in the current position or not.

While dragging a panel in Dashboard layout the following dragging events will be triggered,

- [dragStart](#) - Triggers when panel drag starts
- [drag](#) - Triggers when panel is being dragged
- [dragStop](#) - Triggers when panel drag stops

The following sample demonstrates dragging and pushing of panels. For example, while dragging the panel 0 over panel 1, these panels get collided and push the panel 1 towards the feasible direction, so that the panel 0 gets placed in the panel 1 position.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- Dashboardlayout element declaration -->

@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model
.cellSpacing).Panels(Panel =>
    {

Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div>0</div>").Add();
Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div>1</div>").Add();
Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div>2</div>").Add();
Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div>3</div>").Add();
Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div>4</div>").Add();
Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div>5</div>").Add();
Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div>6</div>").Add();

}).Drag("onDrag").DragStart("onDragStart").DragStop("onDragStop").Render()
    </div>
</div>
<!-- end of dashboardlayout element -->
<script>
function onDrag(args) {
    console.log("Dragging");
}
function onDragStart(args) {
    console.log("Drag Start");
}
function onDragStop(args) {
    console.log("Drag Stop");
}
</script>
<style>
    /* DashboardLayout element styles */
    #dashboard_layout .e-panel .e-panel-container {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>
```

```
}  
</style>
```

HOMECONTROLLER.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using Microsoft.AspNetCore.Mvc;  
namespace WebApplication.Controllers  
{  
    public class HomeController : Controller  
    {  
        public class spacingModel  
        {  
            public double[] cellSpacing { get; set; }  
        }  
        public ActionResult Index()  
        {  
            spacingModel modelValue = new spacingModel();  
            modelValue.cellSpacing = new double[] { 10, 10 };  
            return View(modelValue);  
        }  
    }  
}
```



Customizing the dragging handler

Initially, the complete panel will act as a handler for dragging the panel such that the dragging action occurs on clicking anywhere over a panel. However, this dragging handler for the panels can be customized using the [draggableHandle](#) property to restrict the dragging action within a particular element in the panel.

The following sample demonstrates customizing the dragging handler of the panels where the dragging action of panel occurs only with the header of the panel.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div>
    <!-- Dashboardlayout element declaration -->

    @Html.EJS().DashboardLayout("dashboard_default").Columns(6).CellSpacing(Mode
    1.cellSpacing).DraggableHandle(".e-panel-header").Panels(Panel =>
        {
            Panel.Id("Panel1").SizeX(3).SizeY(2).Row(0).Col(3).Header("<div
            class='header'>Last year sales comparison</div><span class='handler e-icons
            burg-icon'></span>").Content("#column").Add();
            Panel.Id("Panel2").SizeX(3).SizeY(2).Row(0).Col(3).Header("<div
            class='header'>Mobile browsers usage</div><span class='handler e-icons burg-
            icon'></span>").Content("#piel").Add();
            Panel.Id("Panel3").SizeX(3).SizeY(2).Row(1).Col(0).Header("<div
            class='header'>Sales increase percentage</div><span class='handler e-icons
            burg-icon'></span>").Content("#line").Add();
        }).Render()
    </div>
    <!-- end of dashboardlayout element -->
    <div id="column">
        <!-- Chart element declaration -->
        @Html.EJS().Chart("columnChart").Series(series =>
            {
                series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column)
                .XName("month")
                .YName("sales")
                .DataSource(ViewBag.chartSource).Add();
            }).PrimaryXAxis(px =>
px.ValueType(Syncfusion.EJ2.Charts.ValueType.Category)).Height("162px").Rend
er()
        </div>
        <div id="line">
            <!-- Chart element declaration -->
            @Html.EJS().Chart("lineChart").Series(series =>
                {
                    series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
                    .XName("x")
                    .YName("y")
                    .DataSource(ViewBag.lineSource).Add();
                }).Height("162px").Render()
            </div>
            <div id="piel">
                <!-- Chart element declaration -->
                @Html.EJS().AccumulationChart("pieChart1").Series(series =>
                    {
```



```

series.DataLabel(dl => dl.Visible(true).Name("text").Font(ft =>
ft.FontWeight("600")).Position(Syncfusion.EJ2.Charts.AccumulationLabelPositi
on.Inside))

.XName("x")

.YName("y")

.Name("Browser")

.Radius("70%")

.DataSource(ViewBag.pieSource1).Add();

})

.EnableAnimation(false)

.EnableSmartLabels(true)

.Height("162px")

.LegendSettings(ls => ls.Visible(false))

.Tooltip(tp
=> tp.Enable(true).Format("${point.x} : <b>${point.y}%</b>")).Render()
)
</div>
<style>
    /* DashboardLayout element styles */
    .e-dashboard-layout.e-control .e-panel .e-panel-container .e-panel-
header {
        color: rgba(0, 0, 0, 0.61);
    }
    .e-panel .e-header-text {
        padding: 12px 0 12px 0;
    }
    #dashboard .e-panel-container .e-panel-header {
        border-bottom: 1px solid #888383;
    }
    .e-dashboard-layout.e-control .e-panel:hover {
        border: 0px;
    }
    .e-dashboard-layout.e-control .e-panel .e-panel-header {
        font-size: 15px;
        font-weight: 500;
        height: 37px;
        padding: 10px;
        vertical-align: middle;
        /* text-align: left; */
        border-bottom: 0.5px solid rgba(0, 0, 0, .125);
    }
    .e-panel-header {
        padding: 10px;
        margin-bottom: 0;
        background-color: rgba(0, 0, 0, .03);
    }
    .e-dashboard-layout.e-control .e-panel .e-panel-header {

```

```
height: 30px
}
.e-dashboard-layout.e-control .e-panel .e-panel-header div {
  text-align: center;
}
.header {
  padding: 5px;
  display: inline-block;
}
.e-panel-content {
  height: 162px;
}
/* custom generated icons styles */
@@font-face {
  font-family: 'e-icons';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQYukCgIAAAIkAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAAQAAIb718
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAQAIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQpQAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAgAAAAAAMAAUAAMAAQA
AABQABABeAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAGABWAKAAKABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAAAA+oD6gALAAATCQEXCQEHQCEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBWYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEACFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICakKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUEXIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEHERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTEuUUFhUWFxYWFxEQEBApDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMFBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwmEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAgMDAwQE0AUEAWMDAgEBAgMDAwQ
FfAUEAWMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRochR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECawMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAWMDAgEBAgMDAwQFAwQEAWQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxyVExEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHBoZFxyVExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAEAOAQcuAsc+ATceAQEHBgcJgYPAQYWHwE
GFBCHDgEfAR4BPWEWHwEeATsBMjY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBLIYMhhSBhIIrgg
FAzIfe4QJDADwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOEQqWMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4za+v+Ff4VHwHMAc0AAAAAAQAAAAADAQpQAUAUAAEXCQEHAQLHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUFHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFbQTEuXITEREREBAPDw00DAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA
QERERERUUFbQTEuXITEREREBAPDw00DAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcyFhYWFQRUEXIREQ8PDg0MCwoJCAcFBQMCAGMFBQcICQoLDA00Dw8
RERITFBQVFhYWGBCYGRkZGhoZGRkYFxfGfYhYVFBQTEHERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
```

```

RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehM
TFBQUFRoZGRkYFwgWfHYVFBQTEHERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WfHYFwgZGRkaGhkZGRgXGBYWFhUUFBMSEREPdw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAAAABAAAAAAAAABAA0AAQABAAAAAAAAACAAcAdgABAAAAAA
DAA0AFQABAAAAAAAAEAA0AIgABAAAAAAFAAsALwABAAAAAAAGAA0AogABAAAAAAAKACwArwABAAA
AAAAABIAcWADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWlj25zLWl1dHJvUmVndWxhcmtUtaWNvbnMtbnV0cm91LWlj25zLWl1dHJvVmVyc2lvbiA
xLjBlLWlj25zLWl1dHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAAC4AMABlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgBvAEYAbwBuAHQ
AIAbnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAB
NAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAGEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMtbnVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbilzZWZ0CElUX0NsZWZyDElUX0p1bmmttYWlscwR
zdG9wAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.handler.burg-icon:before {
    content: '\e10d';
    font-size: 16px;
}
.handler.burg-icon {
    float: right;
    padding-top: 2%;
}
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public class ChartData
        {
            public string month;
            public double sales;
        }
        public class LineData
        {

```

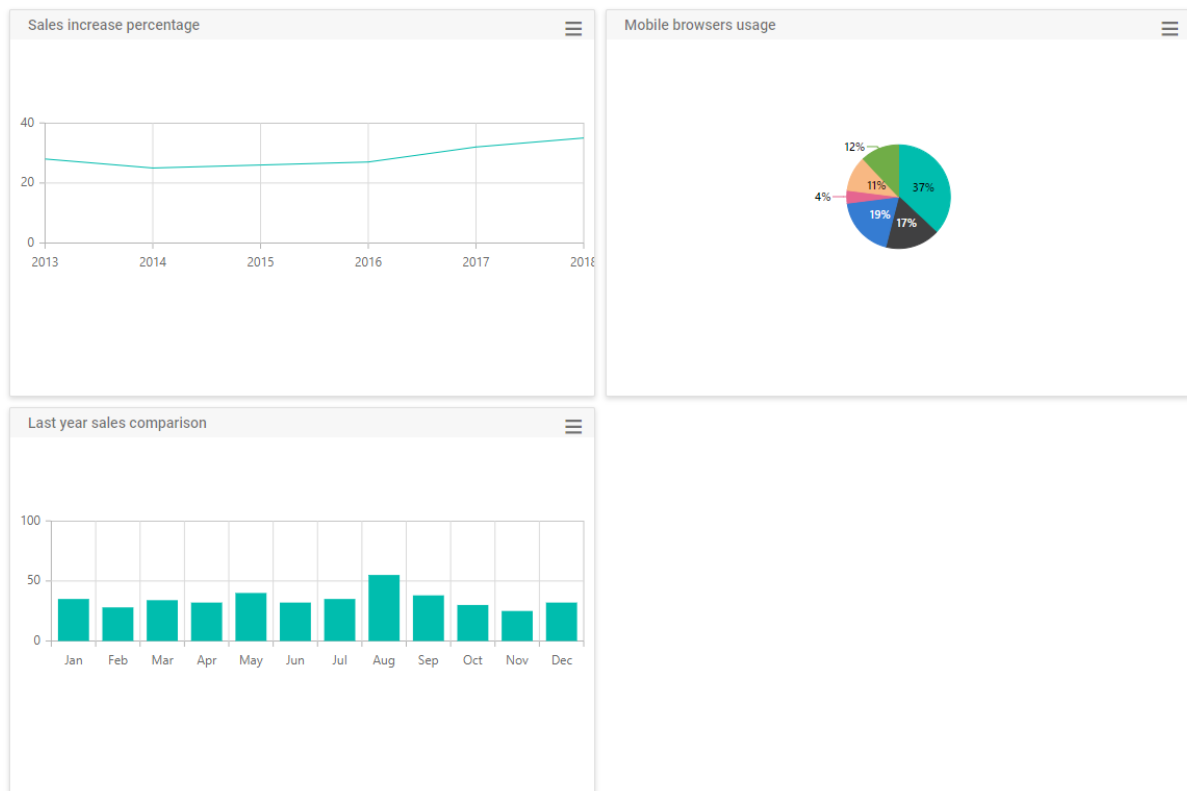
```
        public double x;  
        public double y;  
    }  
    public class PieData  
    {  
        public string x;  
        public double y;  
        public string text;  
    }  
    public ActionResult Index()  
    {  
        List<ChartData> chartData = new List<ChartData>  
        {  
            new ChartData {  
                month = "Jan",  
                sales = 35,  
            },  
            new ChartData {  
                month = "Feb",  
                sales = 28,  
            },  
            new ChartData {  
                month = "Mar",  
                sales = 34,  
            },  
            new ChartData {  
                month = "Apr",  
                sales = 32,  
            },  
            new ChartData {  
                month = "May",  
                sales = 40,  
            },  
            new ChartData {  
                month = "Jun",  
                sales = 32,  
            },  
            new ChartData {  
                month = "Jul",  
                sales = 35,  
            },  
            new ChartData {  
                month = "Aug",  
                sales = 55,  
            },  
            new ChartData {  
                month = "Sep",  
                sales = 38,  
            },  
            new ChartData {  
                month = "Oct",  
                sales = 30,  
            },  
            new ChartData {  
                month = "Nov",  
                sales = 25,  
            },  
        }  
    }  
}
```

```
        new ChartData {
            month = "Dec",
            sales = 32,
        }
    };
    ViewBag.chartSource = chartData;
    List<LineData> lineData = new List<LineData>
    {
        new LineData {
            x = 2013,
            y = 28,
        },
        new LineData {
            x = 2014,
            y = 25,
        },
        new LineData {
            x = 2015,
            y = 26,
        },
        new LineData {
            x = 2016,
            y = 27,
        },
        new LineData {
            x = 2017,
            y = 32,
        },
        new LineData {
            x = 2018,
            y = 35,
        }
    };
    ViewBag.lineSource = lineData;
    List<PieData> pieData1 = new List<PieData>
    {
        new PieData {
            x = "Chrome",
            y = 37,
            text = "37%",
        },
        new PieData {
            x = "UC Browser",
            y = 17,
            text = "17%",
        },
        new PieData {
            x = "iPhone",
            y = 19,
            text = "19%",
        },
        new PieData {
            x = "Others",
            y = 4,
            text = "4%",
        },
        new PieData {
```

```

        x = "Opera",
        y = 11,
        text = "11%",
    },
    new PieData {
        x = "Android",
        y = 12,
        text = "12%",
    }
};
ViewBag.pieSource1 = pieData1;
spacingModel.modelValue = new spacingModel();
modelValue.cellSpacing = new double[] { 10, 10 };
return View(modelValue);
}
}
}

```



Disable dragging of panels

By default, the dragging of panels is enabled in Dashboard Layout. It can also be disabled with the help of [allowDragging](#) API. Setting [allowDragging](#) to false disables the dragging functionality in Dashboard Layout.

CSHTML

```

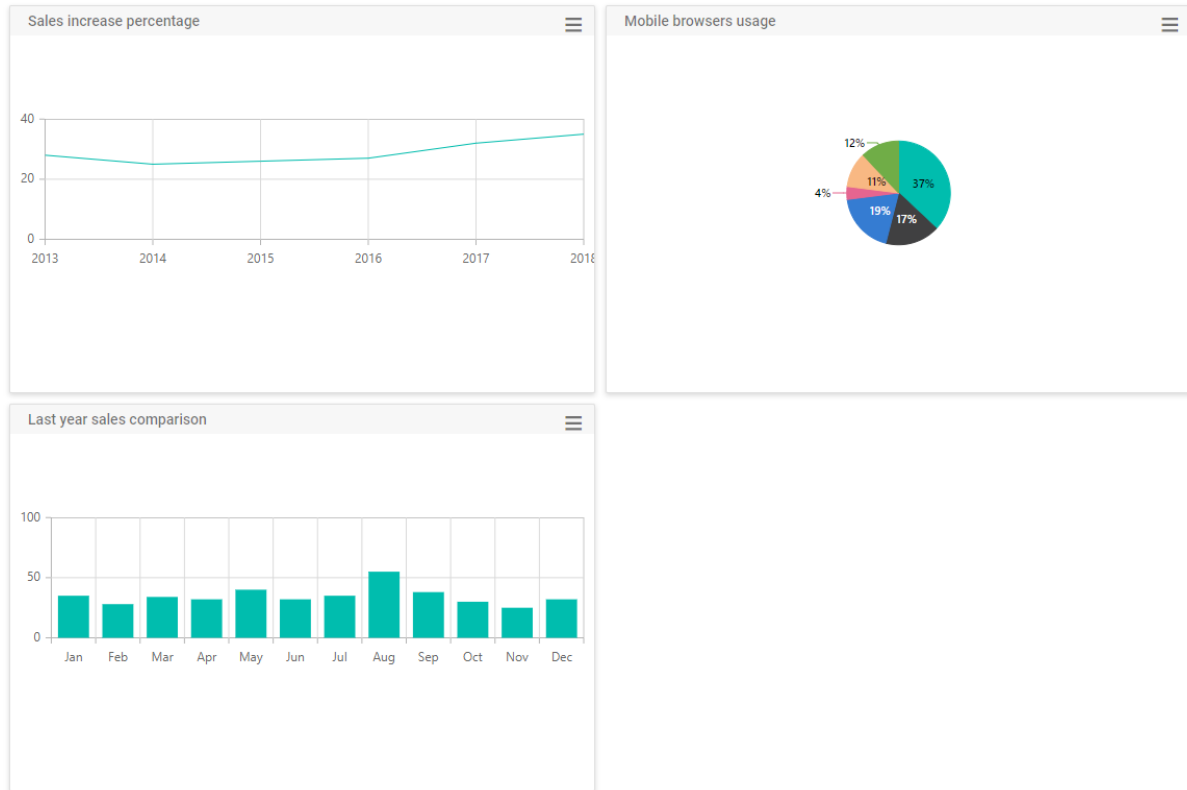
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>

```

```
@Html.EJS().DashboardLayout("dashboard_layout").Columns(5).AllowDragging(false).CellSpacing(Model.cellSpacing).Panels(Panel =>
{
    Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div>0</div>").Add();
    Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div>1</div>").Add();
    Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div>2</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div>3</div>").Add();
    Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div>4</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div>5</div>").Add();
    Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div>6</div>").Add();
}).Render()
</div>
</div>
<style>
    #dashboard_layout .e-panel .e-panel-container {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}
```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Moving of panels programatically

Other than drag and drop, it is possible to move the panels in Dashboard Layout programatically. This can be achieved using `movePanel` method. The method is invoked as follows,

```
`js
movePanel(id, row, col)
`
```

Where,

- id - ID of the panel which needs to be moved.
- row - New row position for moving the panel.
- col - New column position for moving the panel.

Each time a panel's position is changed (Programatically or through UI interaction), the Dashboard Layout's [change](#) event will be triggered.

The following sample demonstrates moving a panel programatically to a new position in the Dashboard Layout's [created](#) event.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
```



```

<div class="control-section">
    <div>

@Html.EJS().DashboardLayout("dashboard_default").Columns(5).Created("onCreated").Change("onChange").CellSpacing(Model.cellSpacing).AllowResizing(true).Panels(Panel =>
    {
        Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div class='content'>0</div>").Add();
        Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div class='content'>1</div>").Add();
        Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div class='content'>2</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div class='content'>3</div>").Add();
        Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div class='content'>4</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div class='content'>5</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div class='content'>6</div>").Add();
    }).Render()
    </div>
</div>
<script>
function onCreated(args) {
    // movePanel("id", row, col)
    this.movePanel("layout_0", 1, 0);
}
//Dashboard Layout's change event function
function onChange(args) {
    console.log("Change event triggered");
}
</script>
<style>
    #dashboard_default .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel

```

```

    {
        public double[] cellSpacing { get; set; }
        public string[] handles { get; set; }
    }
    public ActionResult Index()
    {
        spacingModel modelValue = new spacingModel();
        modelValue.cellSpacing = new double[] { 10, 10 };
        return View(modelValue);
    }
}

```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Resizing of panels

The DashboardLayout component is also provided with the panel resizing functionality which can be enabled or disabled using the [allowResizing](#) property. This functionality allows you to resize the panels dynamically through UI interactions using the resizing handlers, which controls the panel resizing in various directions.

Initially, the panels can be resized only in south-east direction. However, panels can also be resized in east, west, north, south, and south-west directions by defining the required directions with the [resizableHandles](#) property.

On resizing a panel in Dashboard layout the following events will be triggered,

- [resizeStart](#) - Triggers when panel resize starts
- [resize](#) - Triggers when panel is being resized
- [resizeStop](#) - Triggers when panel resize stops

CSSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div className="control-section">
    <div>
        <!-- DashboardLayout element declaration -->

@Html.EJS().DashboardLayout("dashboard_default").Columns(5).ResizableHandles
(Model.handles).AllowResizing(true).Resize("onResize").ResizeStart("onResize
Start").ResizeStop("onResizeStop").CellSpacing(Model.cellSpacing).Panels(Pan
el =>
    {
        Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div
class='content'>0</div>").Add();
        Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div
class='content'>1</div>").Add();
        Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div
class='content'>2</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div
class='content'>3</div>").Add();
        Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div
class='content'>4</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div
class='content'>5</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div
class='content'>6</div>").Add();
    }).Render()
    <!-- end of dashboardlayout element -->
    </div>
</div>
<script>
function onResize(args) {
    console.log("Resizing");
}
function onResizeStart(args) {
    console.log("Resize start");
}
function onResizeStop(args) {
    console.log("Resize Stop");
}
</script>
<style>
    /* DashboardLayout element styles */
    #dashboard_default .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>
```

HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
            public string[] handles { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            modelValue.handles = new string[] { "e-south-east", "e-east",
            "e-west", "e-south-west", "e-south" };
            return View(modelValue);
        }
    }
}
```

**Resizing panels programmatically**

The Dashboard Layout panels can also be resized programmatically by using `resizePanel` method. The method is invoked as follows,

```
`js
```

```
resizePanel(id, sizeX, sizeY)
```

```
,
```

Where,

- [id](#) - ID of the panel which needs to be resized.
- [sizeX](#) - New panel width in cells count for resizing the panel.
- [sizeY](#) - New panel height in cells count for resizing the panel.

The following sample demonstrates resizing panels programatically in the Dashboard Layout's [created](#) event.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>

@Html.EJS().DashboardLayout("dashboard_default").Columns(5).CellSpacing(Model
1.cellSpacing).AllowResizing(true).Created("onCreated").Panels(Panel =>
    {
        Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div
class='content'>0</div>").Add();
        Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div
class='content'>1</div>").Add();
        Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div
class='content'>2</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div
class='content'>3</div>").Add();
        Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div
class='content'>4</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div
class='content'>5</div>").Add();
        Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div
class='content'>6</div>").Add();
    }).Render()
    </div>
</div>
<script>
//Dashboard Layout's created event function
function onCreated(args) {
    // resizePanel("id", sizeX, sizeY)
    this.resizePanel("layout_4", 1, 1);
    this.resizePanel("layout_5", 2, 1);
}
</script>
<style>
#dashboard_default .e-panel .e-panel-container .content {
    vertical-align: middle;
    font-weight: 600;
    font-size: 20px;
    text-align: center;
    line-height: 90px;
}
```

```
</style>
```

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}
```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Floating panels

The floating functionality of the component allows to effectively use the entire layout for the panel's placement. If the floating functionality is enabled, the panels within the layout get floated upwards automatically to occupy the empty cells available in previous rows. This functionality can be enabled or disabled using the [allowFloating](#) property of the component.

The following sample demonstrates how to enable or disable the floating of panels in the DashboardLayout component using [allowFloating](#) property.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div>
    <!-- Button element declaration -->
    @Html.EJS().Button("toggle").Content("Enable Floating").CssClass("e-flat
e-primary e-outline").IsToggle(true).Render()
    <!-- end of button element -->
</div>
<div class="control-section" style="padding-top: 15px;">
    <div>
        <!-- DashboardLayout element declaration -->

        @Html.EJS().DashboardLayout("dashboard_default").Columns(6).AllowFloating(fa
lse).CellSpacing(Model.cellSpacing).Panels(Panel =>
            {
                Panel.SizeX(2).SizeY(2).Row(1).Col(0).Content("<div
class='content'>0</div>").Add();
                Panel.SizeX(2).SizeY(2).Row(2).Col(2).Content("<div
class='content'>1</div>").Add();
                Panel.SizeX(2).SizeY(2).Row(3).Col(4).Content("<div
class='content'>2</div>").Add();
            }).Render()
        <!-- end of dashboardlayout element -->
    </div>
</div>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        // Create instances for dashboardlayout
        var dashboard =
document.getElementById('dashboard_default').ej2_instances[0];
        var resetPanels = dashboard.serialize();
        resetPanels[0].content = '<div class="content">0</div>';
        resetPanels[1].content = '<div class="content">1</div>';
        resetPanels[2].content = '<div class="content">2</div>';
        // Toggle button to enable and disable floating
        document.getElementById('toggle').onclick = function (e) {
            var toggleBtn =
document.getElementById('toggle').ej2_instances[0];
            let panels = [];
            if (toggleBtn.content == "Disable Floating and Reset") {
                toggleBtn.content = 'Enable Floating';
                dashboard.allowFloating = false;
                dashboard.panels = resetPanels;
            } else {
                toggleBtn.content = 'Disable Floating and Reset';
                dashboard.allowFloating = true;
            }
        }
    });
</script>
```

```

    }
    });
});
</script>
<style>
    /* DashboardLayout element styles */
    #dashboard_default .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>

```

HOMECONTROLLER.CS

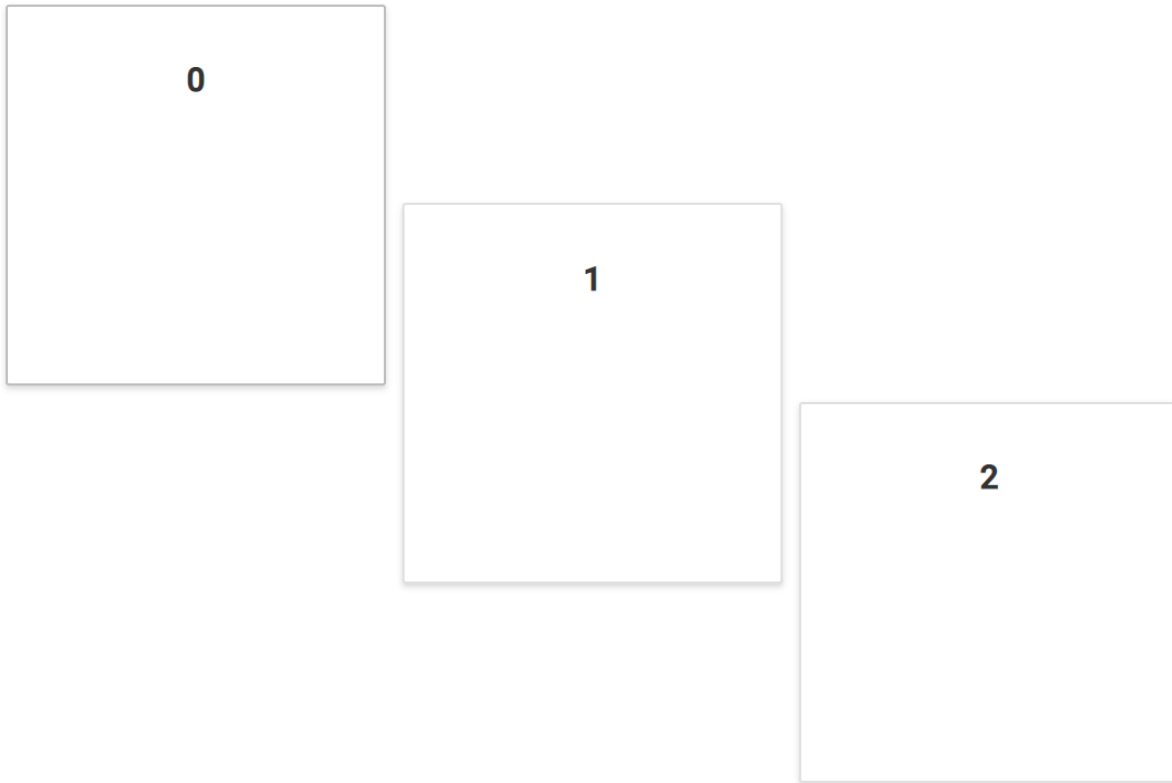
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 10, 10 };
            return View(modelValue);
        }
    }
}

```

When [allowFloating](#) value set as false, the output is like the below.

ENABLE FLOATING



When [allowFloating](#) value set as true, the output is like the below.

DISABLE FLOATING AND RESET



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Responsive and adaptive layout

The control is provided with built-in responsive support, where panels within the layout get adjusted based on their parent element's dimensions to accommodate any resolution which relieves the burden of building responsive dashboards.

The dashboard layout is designed to automatically adapt with lower resolutions by transforming the entire layout into a stacked one, so that the panels will be displayed in a vertical column. By default, whenever the screen resolution meets 600px or lower resolutions this layout transformation occurs. This transformation can be modified for any user defined resolution by defining the [mediaQuery](#) property of the component.

The following sample demonstrates the usage of the [mediaQuery](#) property to turn out the layout into a stacked one in user defined resolution. Here, whenever the window size reaches 700px or lesser, the layout becomes a stacked layout.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div class="control-section">
    <div>
        <!-- Dashboardlayout element declaration -->

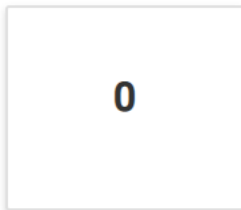
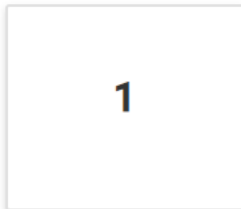
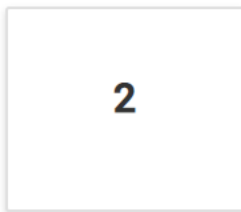
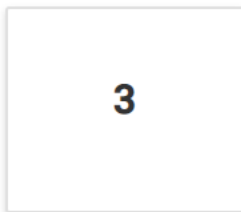
        @Html.EJS().DashboardLayout("dashboard_layout").Columns(5).CellSpacing(Model
        .cellSpacing).MediaQuery("max-width: 700px").Panels(Panel =>
        {

            Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div>0</div>").Add();
            Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div>1</div>").Add();
            Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div>2</div>").Add();
            Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div>3</div>").Add();
        }).Render()
        <!-- end of dashboardlayout element -->
    </div>
</div>
<style>
    /* DashboardLayout element styles */
    #dashboard_layout .e-panel .e-panel-container {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>
```

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
```

```
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 20, 20 };
            return View(modelValue);
        }
    }
}
```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Panel state maintenance

The current layout structure of the Dashboard Layout component can be obtained and saved to construct another dashboard with same panel structure using the `serialize` public method of the component. This method returns the component's current panel setting which can be used to construct a dashboard with the same layout settings.

The following sample demonstrates how to save and restore the state of the panels using the `serialize` method. Click Save to store the panel's settings and click Restore to restore the previously saved panel settings.

CSHTML

```
@model WebApplication.Controllers.HomeController.spacingModel
<div>
    <!-- Button element declaration -->
    @Html.EJS().Button("save").Content("Save").CssClass("e-
primary").Render()
    @Html.EJS().Button("restore").Content("Restore").CssClass("e-flat e-
outline").Render()
    <!-- end of button element -->
</div>
<div class="control-section" style="padding-top: 15px;">
    <div>
        <!-- DashboardLayout element declaration -->

        @Html.EJS().DashboardLayout("dashboard_default").Columns(5).CellSpacing(Mode
l.cellSpacing).Panels(Panel =>
            {
                Panel.SizeX(1).SizeY(1).Row(0).Col(0).Content("<div
class='content'>0</div>").Add();
                Panel.SizeX(3).SizeY(2).Row(0).Col(1).Content("<div
class='content'>1</div>").Add();
                Panel.SizeX(1).SizeY(3).Row(0).Col(4).Content("<div
class='content'>2</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(1).Col(0).Content("<div
class='content'>3</div>").Add();
                Panel.SizeX(2).SizeY(1).Row(2).Col(0).Content("<div
class='content'>4</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(2).Col(2).Content("<div
class='content'>5</div>").Add();
                Panel.SizeX(1).SizeY(1).Row(2).Col(3).Content("<div
class='content'>6</div>").Add();
            }).Created("onCreated").Render()
        <!-- end of dashboardlayout element -->
    </div>
</div>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        // Create instances for dashboardlayout element
        var dashboard =
document.getElementById('dashboard_default').ej2_instances[0];
        // Save the current panels
        document.getElementById('save').onclick = function (e) {
            onCreated();
        };
        // Restore the initial panels
```

```

        document.getElementById('restore').onclick = function (e) {
            dashboard.panels = restoreModel;
        };
    });
    function onCreate() {
        var dashboard =
document.getElementById('dashboard_default').ej2_instances[0];
        restoreModel = dashboard.serialize();
        restoreModel[0].content = '<div class="content">0</div>';
        restoreModel[1].content = '<div class="content">1</div>';
        restoreModel[2].content = '<div class="content">2</div>';
        restoreModel[3].content = '<div class="content">3</div>';
        restoreModel[4].content = '<div class="content">4</div>';
        restoreModel[5].content = '<div class="content">5</div>';
        restoreModel[6].content = '<div class="content">6</div>';
    }
</script>
<style>
    /* DashboardLayout element styles */
    #dashboard_default .e-panel .e-panel-container .content {
        vertical-align: middle;
        font-weight: 600;
        font-size: 20px;
        text-align: center;
        line-height: 90px;
    }
</style>

```

HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        public class spacingModel
        {
            public double[] cellSpacing { get; set; }
        }
        public ActionResult Index()
        {
            spacingModel modelValue = new spacingModel();
            modelValue.cellSpacing = new double[] { 20, 20 };
            return View(modelValue);
        }
    }
}

```



Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the dashboard layout panel header

Use the following CSS to customize the dashboard layout panel header.

```
`css
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-panel-header {
color: #754131;
background-color: #c9e2f7;
text-align: center;
}
```

Customizing the dashboard layout panel content

Use the following CSS to customize the dashboard layout panel content.

```
`css
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-panel-content {
```

```
background-color: #c9e2f7;
padding: 50px;
}
`
```

Customizing the dashboard layout panel resize icon

Use the following CSS to customize the dashboard layout resize icon.

```
`css
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-resize.e-double{
color: #0378d5;
font-size: 30px;
height: 20px;
width: 20px;
}
`
```

Customizing the dashboard layout panel background

Use the following CSS to customize the dashboard layout panel background.

```
`css
.e-dashboardlayout.e-control.e-responsive {
background: #b3d3ed;
}
`
```

Note: You can refer to our [ASP.NET Core Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET Core Dashboard Layout example](#) to know how to present and manipulate data.

Accessibility in ASP.NET MVC Dashboard Layout component

The Dashboard Layout component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dashboard Layout component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Dashboard Layout component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Dashboard Layout component:

| Attributes | Purpose |

| --- | --- |

| `role=list` | Indicates the role as a list for the Dashboard Layout element. |

| `role=listitem` | Indicates the role as a listitem for the Dashboard panels. |

| `role=presentation` | Indicates the role as a presentation for the table when the `showGridLines` property is enabled. |

| `aria-grabbed` | When the panel is chosen for dragging, the `aria-grabbed` attribute is set to "true." If it's set to "false," the element can be grabbed for drag-and-drop, but it won't be actively held. |

Keyboard interaction

Keyboard support is not applicable for the Dashboard Layout.

Ensuring accessibility

The Dashboard Layout component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dashboard Layout component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dashboard Layout component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

DatePicker

Getting Started with ASP.NET MVC DatePicker Control

This section briefly explains about how to include [ASP.NET MVC DatePicker](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

<add namespace="Syncfusion.EJ2"/>

</namespaces>

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

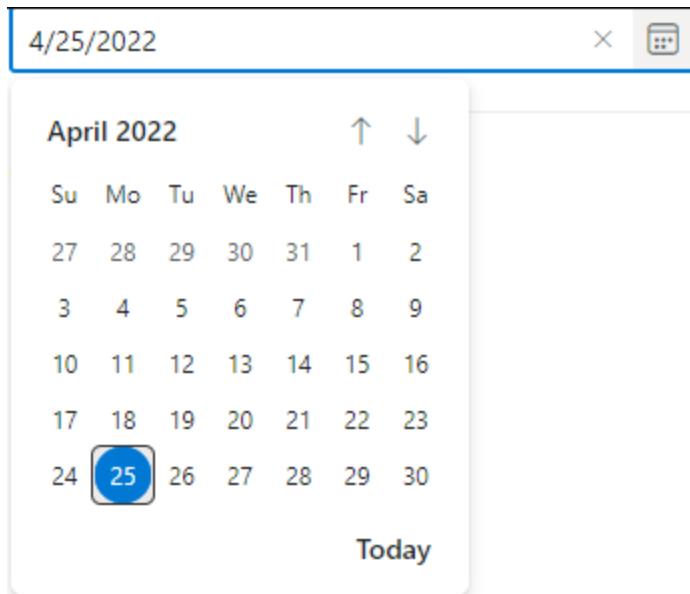
Add ASP.NET MVC DatePicker control

Now, add the Syncfusion ASP.NET MVC DatePicker control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DatePicker control will be rendered in the default web browser.



Note: Running the above code will display the basic DatePicker on the browser.

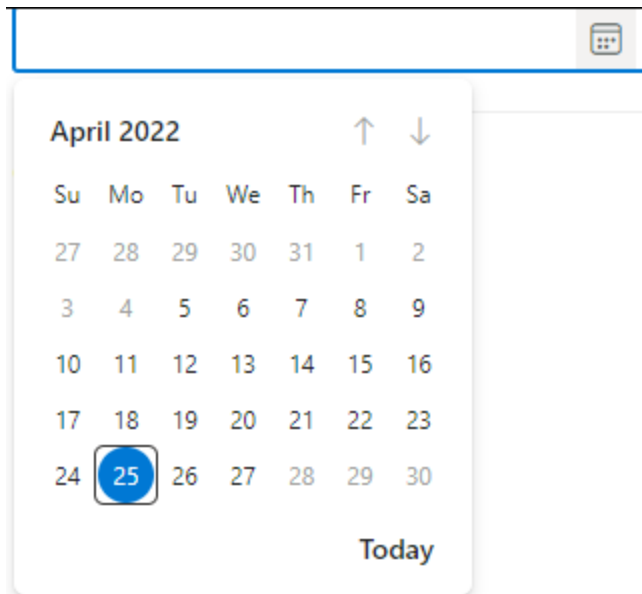
Setting the value within min and max dates

To restrict the selection of date within a specified range, use the [Min](#) and [Max](#) properties.

The below example demonstrates the DatePicker to select a date within a range from 5 to 27 in a current month.

C#HTML

```
@{
    .....
    var minDate= new DateTime(DateTime.Now.Year,DateTime.Now.Month,05);
    var maxDate = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 27);
}
@Html.EJS().DatePicker("datepicker").Min(minDate).Max(maxDate).Render()
```



Note: [View Sample in GitHub.](#)

See also

- [Change the format of selected date](#)
- [Render DatePicker with specific culture](#)
- [How to change the initial view of the DatePicker](#)
- [How to achieve validation with DatePicker](#)
- [How to get and set value in DatePickerFor](#)

Date Range

DatePicker provides an option to select a date value within a specified range by using the [min](#) and [max](#) properties. Always the min value has to be lesser than the max value.

When the min and max properties are configured and the selected date value is out-of-range or invalid, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicate the date is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The below example allows selecting a date within the range from 7th to 27th day in a month.

CSHTML

```
@Html.EJS().DatePicker("element").Value(ViewBag.value).Min(ViewBag.minDate).Max(ViewBag.maxDate).Render()
```

DATERANGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class HomeController : Controller
    {
        public ActionResult DateRange()
        {
            ViewBag.value = new
            DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
            ViewBag.minDate= new
            DateTime(DateTime.Now.Year, DateTime.Now.Month, 07);
            ViewBag.maxDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 27);
            return View();
        }
    }
}
```

Note: If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Date Format

Date format is a way of representing the date value in different string format in textbox.

By default the DatePicker's format is based on the culture. You can also set the own custom format by using the [format](#) property.

Note: Once the date format property has been defined, it will be common to all the cultures.

To know more about the date format standards, refer to the [Internationalization Date Format](#) section.

The following example demonstrates the DatePicker with the custom format (yyyy-MM-dd).

CSHTML

```
@Html.EJS().DatePicker("element").Value(ViewBag.value).Format("yyyy-MM-dd").Placeholder("Enter date").Render()
```

DATE-FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult DateRange()
        {
            ViewBag.value = new
            DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
            return View();
        }
    }
}
```

Enable the Masked Input

DatePicker has `enableMask` property that provides the option to enable the built-in date masking support.

CSHTML

```
@Html.EJS().DatePicker("element").EnableMask("true").Render()
```

MASK-INPUT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Sample()
        {
            return View();
        }
    }
}
```

The mask pattern is defined based on the provided date format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| Keys | Actions |

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the date. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion. |

CSHTML

```
<div class="control-section">
    <div class="control_wrapper">
        <div class="pane">
            <div class="tabs-wrap">
                <div class="wrap">
                    // Specifies the masked DatePicker without format
property.
@Html.EJS().DatePicker("element").EnableMask("true").Render()
                </div>
            </div>
            <div class="tabs-wrap">
                <div class="wrap">
                    // Specifies the masked DatePicker with format.
```

```
@Html.EJS().DatePicker("element").EnableMask("true").Format("M/d/yyyy").Render()

</div>
</div>
</div>
</div>
</div>
```

DATE-FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Sample()
        {
            return View();
        }
    }
}
```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default, it takes the full name of date and time co-ordinates such as `day`, `month`, `year`, `hour` etc.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of `L10n` class for mask placeholder values like below.

```
`typescript
//load the locale object to set the localized mask placeholder value
L10n.load({
    'de': {
        'datepicker': { day: 'Tag', month: 'Monat', year: 'Jahr' }
    }
});
`
```

CSHTML

```
<div class="control-section">
    <div class="control_wrapper">
        <div class="pane">
            <div class="tabs-wrap">
                <div class="wrap">
```

```

// Specifies the masked DatePicker with format
property.
@Html.EJS().DatePicker("element").EnableMask("true").Render()
    </div>
</div>
<div class="tabs-wrap">
    <div class="wrap">
        // Specifies the masked DatePicker without format.
@Html.EJS().DatePicker("element").EnableMask("true").MaskPlaceholder(new
Syncfusion.EJ2.Calendars.DatePickerMaskPlaceholder {Day = "d", Month = "M",
Year = "y"}).Render()
    </div>
</div>
</div>
</div>
</div>
</div>

```

MASK-PLACEHOLDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        {
            public ActionResult Sample()
            {
                return View();
            }
        }
    }
}

```

Globalization in ASP.NET MVC DatePicker Control

Globalization is the combination of adapting the control to various languages by means of parsing and formatting the date or number [Internationalization](#), and also by adding cultural specific customizations and translating the text [localization](#).

By default, DatePicker date format, week and month names are specific to the **American English** culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the **loadCldr** method to load culture specific CLDR JSON data.

- Set the culture by using the [locale](#) property.

To go with the different culture other than **English**, follow the below steps.

- Install the **CLDR-Data** package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the [CLDR-Data](#) link.

`

```
npm install cldr-data --save
```

`

Once the package installed, you can find the culture specific JSON data under the location **/node_modules/cldr-data**.

In ASP.NET MVC refer the culture files directly from **/node_modules/cldr-data** location.

```
`typescript
```

```
function loadCultureFiles(name) {
    var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
    var loader = ej.base.loadCldr;
    var loadCulture = function (prop) {
        var val, ajax;

        ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-data/main/' +
            name + '/' + files[prop], 'GET', false);
        ajax.onSuccess = function (value) {
            val = value;
        };
        ajax.send();
        loader(JSON.parse(val));
    };
    for (var prop = 0; prop < files.length; prop++) {
        loadCulture(prop);
    }
}
```

`

Note: The **Localization** library allows you to localize default text content of the DatePicker. The DatePicker control has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

| Locale keywords | Text |

| ----- | ----- |

| today | Name of the button to choose Today date. |

| placeholder | Hint to describe expected value in input element. |

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of **L10n** class.

```
`typescript
var L10n = ej.base.L10n;
L10n.load({
  'de': {
    'datepicker': { placeholder: 'Wählen Sie ein Datum',
    today: 'heute'
  }
});
`
```

The following example demonstrates the DatePicker in **German** culture.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        datepicker = document.getElementById('datepicker').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            'de': {
                'datepicker': {
                    placeholder: 'Wählen Sie ein Datum',
                    today: 'heute'
                }
            }
        });
        loadCultureFiles('de');
        datepicker.locale = 'de';
    });
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
            } else {
```

```

        ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
    }
    ajax.onSuccess = function (value) {
        val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
}
}
</script>

```

GLOBALIZATION.CS

Right-To-Left

The DatePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction.

The following code example initialize the DatePicker control in Arabic culture and also explains how to set the localized text to the placeholder using `load` method of `L10n` class.

CSHTML

```

@Html.EJS().DatePicker("datepicker").EnableRtl(true).Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        datepicker = document.getElementById('datepicker').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            'ar': {
                'datepicker': {
                    placeholder: 'اختر تاريخا',
                    today: 'اليوم'
                }
            }
        });
        loadCultureFiles("ar");
        datepicker.locale = 'ar';
    });
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {

```

```

        ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../.../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
    } else {
        ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../.../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
    }
    ajax.onSuccess = function (value) {
        val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
}
}
</script>

```

RTL.CS

Strict Mode

The [strictMode](#) is an act, that allows the user to enter only the valid date within the specified min/max range in textbox. If the date is invalid, then the component will stay with the previous value. Else, if the date is out of range, then the component will set the date to the min/max date.

The following example demonstrates the DatePicker in [strictMode](#) with min/max range of 5th to 25th in a month of May. Here, it allows to enter only the valid date within the specified range. If you are trying to enter the out-of-range value as like 28th of May, then the value will set the max date of 25th May, since the value 28th is greater than max value of 25th. Or else if you are trying to enter the invalid date, then the value will stay with the previous value.

CSHTML

```

@Html.EJS().DatePicker("element").StrictMode(true).Format("dd/MM/yyyy").Value(ViewBag.value).Min(ViewBag.minDate).Max(ViewBag.maxDate).Placeholder("Enter date").Render()

```

STRICT-TRUE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController: Controller
    {
        public ActionResult DefaultFunctionalities()

```

```

    {
        ViewBag.value = new DateTime(2018, 5, 28);
        ViewBag.minDate= new DateTime(2018, 5, 5);
        ViewBag.maxDate = new DateTime(2018, 5, 25);
        return View();
    }
}

```

By default, the DatePicker act in strictMode **false** state allows to enter the invalid or out-of-range date in textbox.

If the date is out-of-range or invalid, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicate the date is out of range or invalid.

The following example demonstrates the **strictMode** as **false**. Here, it allows to enter the valid or invalid value in textbox. If you are entering out-of-range or invalid date value, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicate the date is out of range or invalid.

CSHTML

```

@Html.EJS().DatePicker("element").Format("dd/MM/yyyy").Value(ViewBag.value).
Min(ViewBag.minDate).Max(ViewBag.maxDate).Placeholder("Enter date").Render()

```

STRICT-FALSE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController: Controller
    {
        public IActionResult DefaultFunctionalities()
        {
            ViewBag.value = new DateTime(2018, 5, 28);
            ViewBag.minDate= new DateTime(2018, 5, 5);
            ViewBag.maxDate = new DateTime(2018, 5, 25);
            return View();
        }
    }
}

```

Note: If the value of min or max properties changed through code behind, then you have to update the **value** property to set within the range.

Customization

You can customize the entire appearance of the input element and Calendar by using custom [cssClass](#) property, and also you can use the calendar's [renderDayCell](#) event to customize the appearance of the each day cell.

Below is the list of classes that provides flexible way to customize the DatePicker component.

Class Name	Description
---	---
e-date-wrapper	Applied to DatePicker wrapper
e-datepicker	Applied to the DatePicker element.
e-float-text	Applied to the floating label.
e-date-icon	Applied to the DatePicker icon.
e-popup-wrapper	Applied to DatePicker popup wrapper.
e-calendar	Applied to Calendar element.
e-header	Applied to Calendar header.
e-title	Applied to Calendar title.
e-icon-container	Applied to Calendar previous and next icon container.
e-prev	Applied to Calendar previous icon.
e-next	Applied to Calendar next icon.
e-weekend	Applied to Calendar weekend dates.
e-other-month	Applied to Calendar other month dates.
e-day	Applied to each day cell of the Calendar.
e-selected	Applied to Calendar selected dates.
e-disabled	Applied to Calendar disabled dates.

The following example disables the weekends of every month using `renderDayCell` event. Here, the `e-disabled` class is used to highlight the disabled date.

CSHTML

```
@Html.EJS().DatePicker("element").RenderDayCell("onRenderCell").Placeholder(
    "Choose a Date").Render()
<script>
    function onRenderCell(args) {
        if (args.date.getDay() == 0 || args.date.getDay() == 6) {
            //sets isDisabled to true to disable the date.
            args.isDisabled = true;
            //To know about the disabled date customization, you can refer
            in "styles.css".
        }
    }
</script>
```

CUSTOMIZATION.CS

See Also

- [How to disable the DatePicker control](#)
- [How to set read-only for DatePicker](#)
- [How to customize the DatePicker day header](#)

Start and Depth View

The DatePicker has the following predefined views that provides a flexible way to navigate back and forth to select the date.

View	Description
------	-------------

---	---
-----	-----

month (default)	Displays the days in a month
-----------------	------------------------------

year	Displays the months in a year
------	-------------------------------

decade	Displays the years in a decade
--------	--------------------------------

Start view

You can use the [start](#) property to define the initial rendering view.

The following example demonstrates how to create a DatePicker with **decade** as initial rendering view.

CSHTML

```
@Html.EJS().DatePicker("element").Start(Syncfusion.EJ2.Calendars.CalendarView.Decade).Placeholder("Choose a Date").Render()
```

START.CS

Depth view

Define the [depth](#) property to control the view navigation.

Note: Always the depth view has to be smaller than the start view, otherwise the view restriction will be not restricted.

CSHTML

```
@Html.EJS().DatePicker("element").Depth(Syncfusion.EJ2.Calendars.CalendarView.Year).Start(Syncfusion.EJ2.Calendars.CalendarView.Decade).Placeholder("Choose a Date").Render()
```

DEPTH.CS

Note: To know more about Calendar views refer the Calendar's [Calendar Views](#) section.

Accessibility

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

The DatePicker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DatePicker component is outlined below.

| [Accessibility Criteria](#) | [Compatibility](#) |


| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

DatePicker provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA supports is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied to the input element.

To know about the accessibility of Calendar, refer to the Calendar's [Accessibility](#) section.

It helps to provide information about the widget for assistive technology to the disabled person in screen reader.

- **Aria-expanded:** attributes indicates the state of a collapsible element.
- **Aria-disabled:** attribute indicates the disabled state of this DatePicker control.
- **Aria-activedescendent:** attribute helps in managing the current active child of the DatePicker control.

Keyboard Interaction

You can use the following keys to interact with the DatePicker. The control implements the keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the below list of shortcut keys.

Input Navigation

Before opening the popup, use the below list of keys to control the popup element.

| **Press** | **To do this** |

| --- | --- |

| **Alt + Down Arrow** | Opens the popup. |

| **Alt + Upper Arrow** | Closes the popup. |

| **Esc** | Closes the popup. |

Calendar Navigation

Use the below list of keys to navigate the Calendar after the popup has opened.

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | Focus the previous week date. |

| **Down Arrow** | Focus the next week date. |

| **Left Arrow** | Focus the previous date. |

| **Right Arrow** | Focus the next date. |

| **Home** | Focus the first date in the month. |

| **End** | Focus the last date in the month. |

| **Page Up** | Focus the same date in the previous month. |

| Page Down | Focus the same date in the next month. |

| Enter | Select the currently focused date. |

| Shift + Page Up | Focus the same date in the previous year. |

| Shift + Page Down | Focus the same date in the previous year. |

| Control + Upper Arrow | Moves into the inner level of view like month-year, year-decade |

| Control + Down Arrow | Moves out from the depth level view like decade-year, year-month |

| Control +Home | Focus the starting date in the current year. |

| Control +End | Focus the ending date in the current year. |

Note: To focus the DatePicker control use the alt+t keys.

CSHTML

```
@Html.EJS().DatePicker("element").Placeholder("Choose a Date").Render()  
<script>  
    document.addEventListener('keyup', function (e) {  
        if (e.altKey && e.keyCode === 84) {  
            // press alt+t to focus the control.  
            var datepickerObject =  
document.getElementById("element").ej2_instances[0];  
            datepickerObject.element.focus();  
        }  
    })  
</script>
```

ACCESSIBILITY.CS

Ensuring accessibility

The DatePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DatePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DatePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Style and appearance in DatePicker Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DatePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height and font size /
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {
height: 40px;
font-size: 20px;
}
`
```

Customizing the DatePicker icon element

Use the following CSS to customize the DatePicker icon element.

```
`css
/ To specify background color and font size /
.e-input-group .e-input-group-icon:last-child, .e-input-group.e-control-wrapper .e-input-group-icon:last-child {
font-size: 12px;
background-color: darkgray;
}
`
```

Customizing the Calendar popup of the DatePicker

Check the below section, to customize the style and appearance of the Calendar component.

[Customizing Calendar's style and appearance](#)

Full screen mode support in mobiles and tablets

The DatePicker full-screen mode feature enables users to view the popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DatePicker, simply set the [FullScreenMode](#) API value to `true`. This action will extend the calendar element to occupy the entire screen on mobile devices.

Default Sample

5/7/2018



How To

Disabled State

To disable the DatePicker, use its [enable](#) property.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Enabled(false).Render()
```

DISABLE.CS

Set the placeholder

The following example demonstrates how to set `placeholder` in the DatePicker control.

Using [placeholder](#) you can display a short hint in the input element.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Placeholder("Enter date").Render()
```

PLACEHOLDER.CS

Customize the datepicker day header

You can change the format of the day that has to be displayed in header using [dayHeaderFormat](#) property.

Note: By default, the format is `Short`.

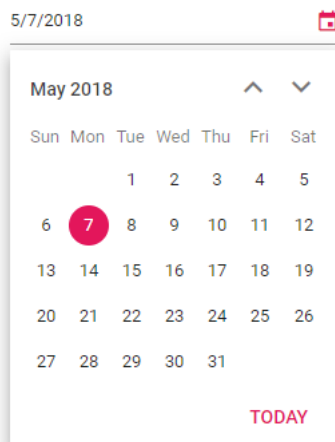
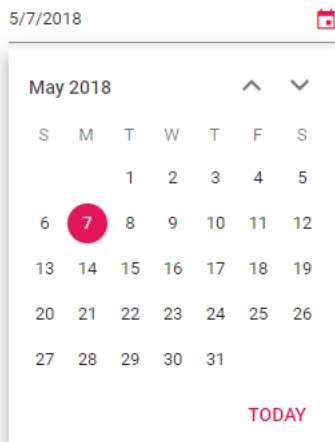
You can find the possible formats on below.

Name	Description
----- -----	
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

CSHTML

```
@Html.EJS().DatePicker("datepicker").DayHeaderFormat(DayHeaderFormat.Short).Render()
```

HEADERFORMAT.CS



Set the readonly

The following example demonstrates how to set `readonly` in DatePicker control. You can achieve this by using `readonly` property.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Readonly(true).Value(ViewBag.value).Placeholder("Choose a Date").Render()
```

READ.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController: Controller
    {
        public ActionResult sample()
        {
        }
    }
}
```

```
{
    ViewBag.value = new
DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
    return View();
}
}
```

Open the DatePicker popup upon focusing input of DatePicker

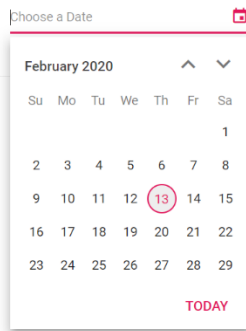
You can open the DatePicker popup on input focus by calling the `show` method in the input `focus` event.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Focus("onFocus").Width("300px").Placeholder("Choose a Date").Render()
<script>
    function onFocus(args) {
        var datepickerObject =
document.getElementById("datepicker").ej2_instances[0];
        datepickerObject.show()
    }
</script>
```

OPENPOPUP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController: Controller
    {
        public ActionResult sample()
        {
            ViewBag.value = new
DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
            return View();
        }
    }
}
```



Prevent the popup close

To prevent the DatePicker popup from closing, use the `preventDefault` method from the `PreventableEventArgs`.

CSHTML

```
@Html.EJS().DatePicker("datepicker").Close("onClose").Placeholder("Choose a Date").Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        var datepickerObject =
document.getElementById("datepicker").ej2_instances[0];
        datepickerObject.show();
    });
    function onClose(args) {
        // prevent the popup close
        args.preventDefault();
    }
</script>
```

POPUPCLOSE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController: Controller
    {
        public ActionResult sample()
        {
            ViewBag.value = new
DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
            return View();
        }
    }
}
```



```
}

```

Client side validation

To achieve the client side validation in a DatePicker control by using [Essential JavaScript 2 FormValidator](#). It provides an option to customize the feedback error messages to the corresponding fields to take action to resolve the issue.

In this below example, the required field validation is implemented by mapping the name attribute value to the rules property. It will validate the DatePicker control and display the validation message when the textbox value is empty during form post back or focus out.

CSHTML

```
<form id="form-element" class="form-vertical">
    <div class="form-group">
        <div class="col-sm-6">

@Html.EJS().DatePicker("datevalue").Value(ViewBag.value).Render()
        </div>
    </div>
</form>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        var options = {
            rules: {
                //must specify the name attribute value in rules section
                'datevalue': { required: true }
            },
            customPlacement: (inputElement, errorElement) => {
                //to place the error message in custom position
                //inputElement - target element where the error text will be
                //errorElement - error text which will be displayed.
                appended
                inputElement.parentElement.parentElement.appendChild(errorElement);
            }
        };
        var formObject = new ej.inputs.FormValidator('#form-element',
options);
    });
</script>

```

VALIDATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController: Controller
    {
        public ActionResult sample()
        {

```

```

        ViewBag.value = new
        DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
        return View();
    }
}

```

Render DatePickerFor

The DatePickerFor component can be rendered by passing a value from the model. The selected date value can be retrieved during form submission using the post method at the server end.

CSHTML

```

@model EJ2CoreSampleBrowser.Controllers.DatePicker
@using (Html.BeginForm())
{
    @Html.EJS().DatePickerFor(model => model.value).Render()
    <div>
        @Html.ValidationMessageFor(model => model.value)
    </div>
    <div id="submitbutton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}

```

DATEPICKERFOR.CS

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class DatePicker
    {
        [Required(ErrorMessage = "Please enter the value")]
        public DateTime? value { get; set; }
    }
    public class HomeController : Controller
    {
        DatePicker DatePickerValue = new DatePicker();
        public ActionResult Index()
        {
            DatePickerValue.value = new DateTime(2020, 03, 03);
            return View(DatePickerValue);
        }
        [HttpPost]
        public ActionResult Index(DatePicker model)
        {
            //posted value is obtained from the model
            DatePickerValue.value = model.value;
            return View(DatePickerValue);
        }
    }
}

```

```

public class HomeController : Controller
{
    DatePicker DatePickerValue = new DatePicker();
    0 references
    public ActionResult Index()
    {
        DatePickerValue.value = new DateTime(2020, 03, 03);
        return View(DatePickerValue);
    }
    [HttpPost]
    0 references
    public ActionResult Index(DatePicker model)
    {
        //posted value is obtained from the model
        ▶ DatePickerValue.value = model.value;
        return View(DatePickerValue);
    }
}

```

model.value | {3/3/2020 12:00:00 AM}

Posting the selected dates to the controller

Post back is the action of posting the data back to the submitting page. In the following example, value for the DatePicker from the **change** event is sent through **Ajax** post to the controller.

In this example, the content type is "application/json" and the request body is a JSON object. In the action method, you can pass a model as the parameter. The **POST** method determines how data is sent from the client via the form, to the server.

CSHTML

```

@model EJ2_MVC.Controllers.FormData
@Html.EJS().DatePickerFor(model =>
model.date).Change("submit").Width("200px").Render()
<script>
    function submit(args) {
        var dateStr = args.value.toUTCString(); // Convert to UTC string
        $.ajax({
            type: "POST",
            url: '@Url.Action("ChangeData")',
            contentType: "application/json", // define content type
            dataType: "json",
            data: JSON.stringify({ 'date': dateStr}), // pass data to
            controller
        })
    }
</script>

```

POSTBACK-SELECTED-DATES-MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc;
namespace EJ2Sample.Controllers
{

```

```
public class FormData
{
    public String date { get; set; }
    public DateTime recievedDate { get; set; }
}
public class HomeController : Controller
{
    // GET: /<controller>/
    public ActionResult Index()
    {
        return View();
    }
    [HttpPost]
    public ActionResult ChangeData(FormData formData)
    {
        //Process using recieved date
        formData.recievedDate = DateTime.Parse(formData.date);
        return Json(formData);
    }
}
```

DateRangePicker

Getting Started with ASP.NET MVC DateRangePicker Control

This section briefly explains about how to include [ASP.NET MVC DateRangePicker](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

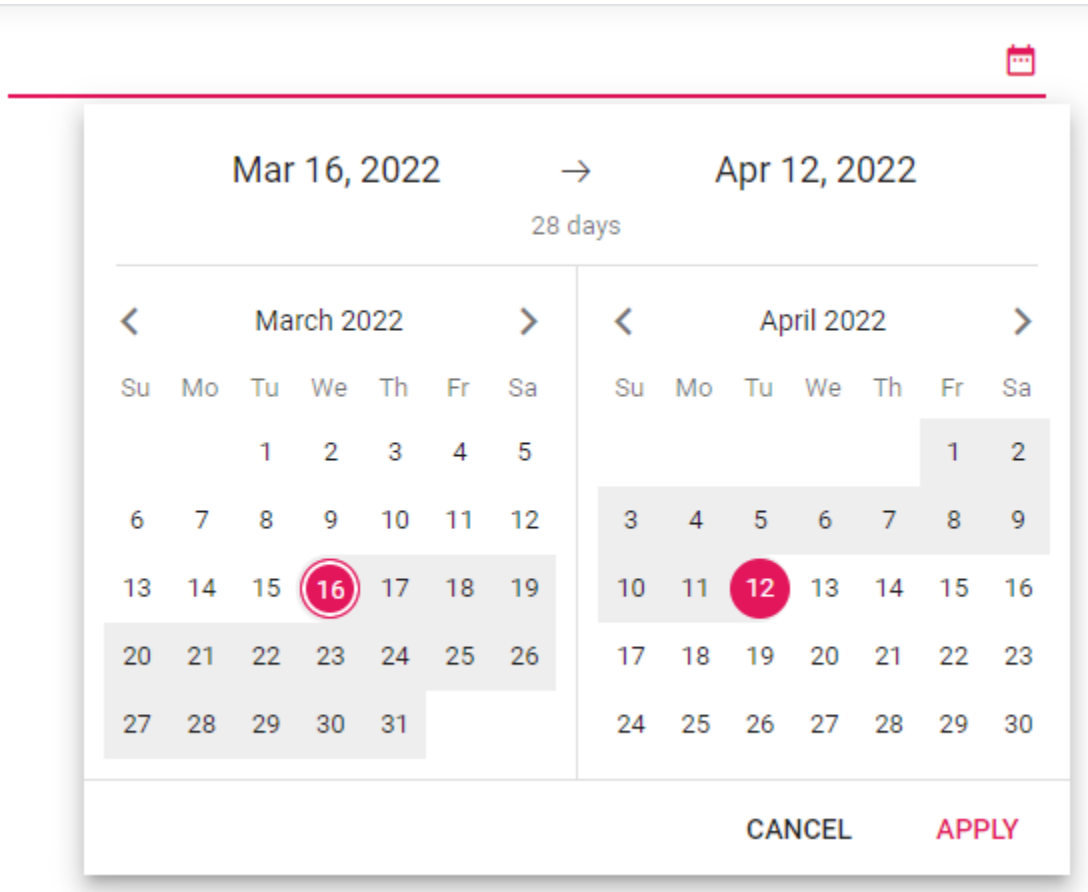
Add ASP.NET MVC DateRangePicker control

Now, add the Syncfusion ASP.NET MVC DateRangePicker control in **~/Views/Home/Index.cshtml** page.

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DateRangePicker control will be rendered in the default web browser.

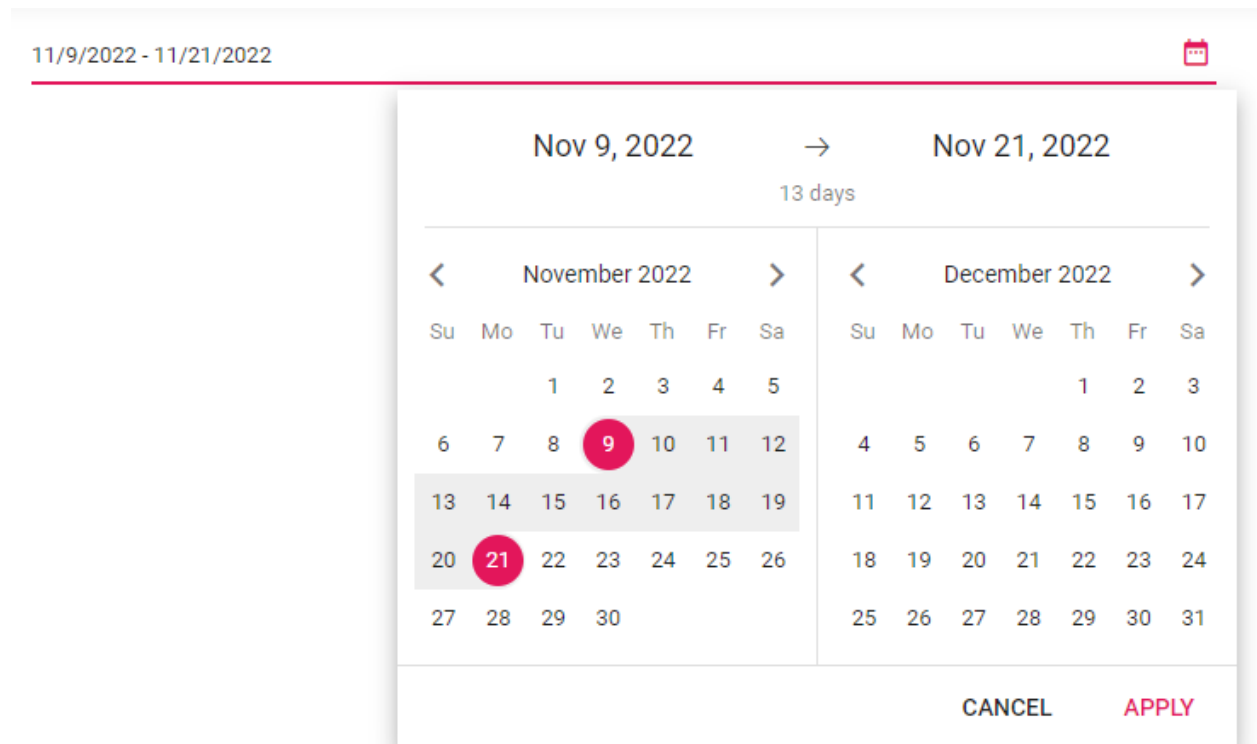


Setting the start and end date

The start and end date in a range can be defined with the help of [StartDate](#) and [EndDate](#) property. The following example demonstrates to set the start and end date on initializing the DateRangePicker.

CSHTML

```
@{
    .....
    var startDate = new DateTime(2022, 11, 09);
    var endDate = new DateTime(2022, 11, 21);
}
@Html.EJS().DateRangePicker("daterangepicker").StartDate(startDate).EndDate(
endDate).Render()
```



Note: [View Sample in GitHub.](#)

See also

- [Render DateRangePicker with range restriction](#)
- [Render DateRangePicker with specific culture](#)
- [How to get and set value in DateRangePickerFor](#)

Range Restriction

Range selection in a DateRangePicker can be made-to-order with desire restrictions based on the application needs.

Restrict the range within a range

You can restrict the minimum and maximum date that can be allowed as start and end date in a range selection with the help of [min](#), [max](#) properties.

- **min** – sets the minimum date that can be selected as startDate.
- **max** – sets the maximum date that can be selected as endDate.

CSHTML

```
@Html.EJS().DateRangePicker("element").Min(ViewBag.minDate).Max(ViewBag.maxDate).Placeholder("Choose a DateRange").Render()
```

DATERANGE.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class DateRangePickerController : Controller
    {
        public ActionResult DateRange()
        {
            ViewBag.minDate= new
            DateTime(DateTime.Now.Year,DateTime.Now.Month,15);
            ViewBag.maxDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month+1, 15);
            return View();
        }
    }
}

```

Note: If the value of min or max property is changed through code behind, update the start date and end date properties to set within the range. Or else, if the start and end date is out of specified date range, a validation error class will be appended to the input element. If strictMode is enabled, and both the start, end date is lesser than the min date then start and end date will be updated with min date. If both the start and end date is higher than the max date then start and end date will be updated with max date. Or else, if startDate is less than min date, startDate will be updated with min date or if endDate is greater than max date, endDate will be updated with the max date.

Range span

Span between ranges can be limited to avoid excess or less days selection towards the required days in a range. In this, minimum and maximum span allowed within the date range can be customized by [minDays](#) and [maxDays](#) properties.

- [minDays](#) - Sets the minimum number of days between start and end date.
- [maxDays](#) - Sets the maximum number of days between start and end date.

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").Placeholder("Select a Range").MinDays(4).MaxDays(10).Render()
```

RANGESPAN.CS

Strict mode

DateRangePicker provides an option to limit the user towards entering the valid date. With strict mode, you can set only the valid date. If any invalid range is specified, the date range value resets to previous value. This restriction can be availed by enabling [strictMode](#) property as true.

CSHTML


```
@Html.EJS().DateRangePicker("element").StrictMode(true).Placeholder("Select a Range").Min(ViewBag.minDate).Max(ViewBag.maxDate).StartDate(ViewBag.startDate).EndDate(ViewBag.endDate).Render()
```

STRICKMODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class DateRangePickerController : Controller
    {
        public ActionResult DateRange()
        {
            ViewBag.minDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 15);
            ViewBag.maxDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month + 1, 15);
            ViewBag.startDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 20);
            ViewBag.endDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month + 1, 25);
            return View();
        }
    }
}
```

Globalization in ASP.NET MVC DateRangePicker Control

Globalization is the combination of adapting the control to various languages by means of parsing and formatting the date or number [Internationalization](#) and also by adding cultural specific customizations and translating the text [localization](#).

By default, DateRangePicker date format, week, and month names are specific to the **American English** culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the **loadCldr** method to load culture specific CLDR JSON data.

- Set the culture by using the [locale](#) property.

To go with the different culture other than **English**, follow the below steps.

- Install the **CLDR-Data** package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the [CLDR-Data](#) link.

```
npm install cldr-data --save
```

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

In ASP.NET MVC refer the culture files directly from `/node_modules/cldr-data` location.

```
`typescript
function loadCultureFiles(name) {
var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
var loader = ej.base.loadCldr;
var loadCulture = function (prop) {
var val, ajax;
ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-data/main/' +
name + '/' + files[prop], 'GET', false);
ajax.onSuccess = function (value) {
val = value;
};
ajax.send();
loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
loadCulture(prop);
}
}
```

Note: The `Localization` library allows you to localize default text content of the DateRangePicker. The DateRangePicker control has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

	Locale keywords		Text	
	-----		-----	
	placeholder		Hint to describe expected value in input element.	
	startLabel		Label to represent the start date.	
	endLabel		Label to represent the end date.	
	applyText		Text present in the apply button.	
	cancelText		Text present in the cancel button.	
	selectedDays		Text to represent selected days.	

| days | Text represents days. |

| customRange | Text present in the custom range button in presets container. |

- Before changing to a culture other than English, ensure that locale text for the concerned culture is loaded through `load` method of `L10n` class.

```
`typescript
var L10n = ej.base.L10n;
L10n.load({
'de': {
'daterangepicker': { placeholder: 'Wählen Sie einen Bereich aus',
startLabel: 'Wählen Sie Startdatum',
endLabel: 'Wählen Sie Enddatum',
applyText: 'Sich bewerben',
cancelText: 'Stornieren',
selectedDays: 'Ausgewählte Tage',
days: 'Tage',
customRange: 'benutzerdefinierten Bereich'
}
}
});
`
```

The following example demonstrates the DateRangePicker in German culture.

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        daterangepicker =
document.getElementById('daterangepicker').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            'de': {
                'daterangepicker': {
                    placeholder: 'Wählen Sie einen Bereich',
                    today: 'heute'
                }
            }
        });
        loadCultureFiles('de');
        daterangepicker.locale = 'de';
    });
    function loadCultureFiles(name) {
```

```

        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
            } else {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
        for (var prop = 0; prop < files.length; prop++) {
            loadCulture(prop);
        }
    }
</script>

```

GLOBALIZATION.CS

Right-To-Left

The DateRangePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew. To display the text in the right-to-left direction, use [enableRtl](#) property.

The code example demonstrates the DateRangePicker control in Arabic culture. It also explains how to set localized text to the placeholder using `L10n.load` method.

The following example demonstrates DateRangePicker in Arabic culture with right-to-left direction.

CSHTML

```

@Html.EJS().DateRangePicker("daterangepicker").EnableRtl(true).Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        daterangepicker =
document.getElementById('daterangepicker').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            'ar': {
                'daterangepicker': {
                    placeholder: 'حدد نطاقا',
                    today: 'اليوم'
                }
            }
        });
    });
</script>

```

```

    }
  }
});
loadCultureFiles("ar");
daterangepicker.locale = 'ar';
});
function loadCultureFiles(name) {
  var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
  if (name === 'ar') {
    files.push('numberingSystems.json');
  }
  var loader = ej.base.loadCldr;
  var loadCulture = function (prop) {
    var val, ajax;
    if (name === 'ar' && prop === files.length - 1) {
      ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
    } else {
      ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
    }
    ajax.onSuccess = function (value) {
      val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
  };
  for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}
</script>

```

RTL.CS

Customization

The DateRangePicker is available for UI customization that can be achieved by using available properties and events in the control.

Day cell format

The DateRangePicker is available for UI customization based on your application requirements. It can be achieved by using [renderDayCell](#) event that provides an option to customize each day cell on rendering.

The following example disables the weekends of every month by using `renderDayCell` event.

CSHTML

```

@Html.EJS().DateRangePicker("element").RenderDayCell("onRenderCell").Placeholder("Select a Range").Render()
<script>

```

```
function onRenderCell(args) {
    if (args.date.getDay() == 0 || args.date.getDay() == 6) {
        //sets isDisabled to true to disable the date.
        args.isDisabled = true;
        //To know about the disabled date customization, you can refer
        in "styles.css".
    }
}
</script>
```

DAYCELL.CS

First day of week

Start day in a week will differ based on the culture, but you can also customize this based on the application needs. For this, you have to make use of [firstDayOfWeek](#) property. By default, first day of a week in en-US is Sunday.

CSHTML

```
@Html.EJS().DateRangePicker("element").FirstDayOfWeek(2).Render()
```

FIRSTDAY.CS

See Also

- [How to customize DateRangePicker using cssClass](#)
- [How to disable DateRangePicker control](#)
- [How to customize the DateRangePicker day header](#)

Accessibility

The DateRangePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DateRangePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

```

| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The web accessibility makes web content and web applications more accessible for people with disabilities. It especially helps in dynamic content change and development of advanced user interface controls with AJAX, HTML, JavaScript, and related technologies. DateRangePicker provides built-in compliance with [WAI-ARIA](#) specifications. WAI-ARIA support is achieved through the attributes like `aria-expanded`, `aria-disabled`, and `aria-activedescendant` applied as an input element.

To know about the accessibility of Calendar, refer to the Calendar's [Accessibility](#) section.

It helps people with disabilities by providing information about the widget for assistive technology in the screen readers. DateRangePicker control contains grid role and grid cell for each day cell.

- **Aria-expanded:** Indicates the currently selected date of the DateRangePicker control.
- **Aria-disabled:** Indicates the disabled state of the DateRangePicker control.

Keyboard Interaction

Use the below keys to interact with the DateRangePicker. This control implements the keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys:

Input Navigation

Before opening the popup, use the following list of keys to control the popup element.

| Press | To do this |

| --- | --- |

| Alt + Down Arrow | Opens the popup. |

| Alt + Upper Arrow | Closes the popup. |

| Esc | Closes the popup. |

Calendar Navigation

Use the following list of keys to navigate the currently focused Calendar after the popup has opened.

| Press | To do this |

| --- | --- |

| Upper Arrow | Focuses the same day of the previous week. |

| Down Arrow | Focuses the same day of the next week. |

| Left Arrow | Focuses the day before. |

| Right Arrow | Focuses the next day. |

| Home | Focuses the first day of the month. |

| End | Focuses the last day of the month. |

| Page Up | Focuses the same date of the previous month. |

| Page Down | Focuses the same date of the next month. |

| Enter | Selects the currently focused date. |

| Shift + Page Up | Focuses the same date for the previous year. |

| Shift + Page Down | Focuses the same date for the next year. |

| Control + Home | Focuses the first date of the current year. |

| Control + End | Focuses the last date of the current year. |

| Alt + Right | Focuses through out the pop-up container in forward direction. |

| Alt + Left | Focuses through out the pop-up container in backward direction. |

Note: To focus the DateRangePicker control use the alt+t keys.

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").Placeholder("Select a
Range").Render()
<script>
    document.addEventListener('keyup', function (e) {
        if (e.altKey && e.keyCode === 84) {
            // press alt+t to focus the control.
        }
    });
</script>
```



```
var daterangepickerObject =  
document.getElementById("daterangepicker").ej2_instances[0];  
    daterangepickerObject.element.focus();  
}  
})  
</script>
```

ACCESSIBILITY.CS

Ensuring accessibility

The DateRangePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DateRangePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DateRangePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Style and appearance in DateRangePicker Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DateRangePicker wrapper element

Use the following CSS to customize the appearance like height and font size of the wrapper element.

`css

/ To specify height and font size /

.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {

font-size: 20px;

height: 40px;

}

`

Customizing the DateRangePicker icon element

Use the following CSS to customize the DateRangePicker icon element

`css

/ To specify background color and font size /

.e-input-group .e-input-group-icon:last-child, .e-input-group.e-control-wrapper .e-input-group-icon:last-child {

background-color: darkgray;

```
font-size: 14px;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar header

Use the following CSS to customize the DateRangePicker popup calendar header

```
`css
```

```
/ To specify background and height /
```

```
.e-daterangepicker.e-popup .e-range-header {
```

```
background: beige;
```

```
height: 80px;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar header title

Use the following CSS to customize the DateRangePicker popup calendar header title

```
`css
```

```
/ To specify color and font size /
```

```
.e-daterangepicker.e-popup .e-range-header .e-start-label, .e-daterangepicker.e-popup .e-range-header  
.e-end-label {
```

```
color: brown;
```

```
font-size: 30px;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar content

Use the following CSS to customize the DateRangePicker popup calendar content

```
`css
```

```
/ To specify background color /
```

```
.e-daterangepicker.e-popup .e-calendar {
```

```
background-color: brown;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar content title

Use the following CSS to customize the DateRangePicker popup calendar content title

```
`css
```

```
/ To specify color and font size /
```

```
.e-daterangepicker.e-popup .e-calendar .e-header .e-title {
color: beige;
font-size: 20px;
}
`css
```

Customizing the DateRangePicker popup calendar previous and next icon

Use the following CSS to customize the DateRangePicker popup calendar previous and next icon

```
`css
/ To specify font size /
.e-calendar .e-header .e-prev, .e-calendar .e-header .e-next, .e-bigger.e-small .e-calendar .e-header .e-
prev, .e-bigger.e-small .e-calendar .e-header .e-next {
font-size: 20px;
}
`css
```

Customizing the DateRangePicker popup calendar date cell grid on hovering

Use the following CSS to customize the DateRangePicker popup calendar date cell grid on hovering

```
`css
/ To specify background color and border /
.e-calendar .e-content td:hover span.e-day {
background-color: beige;
border: 1px solid black;
}
`css
```

Customizing the DateRangePicker popup calendar primary button in footer

Use the following CSS to customize the DateRangePicker popup calendar primary button in footer

```
`css
/ To specify background color and border color /
.e-daterangepicker.e-popup .e-footer .e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-
popup .e-footer .e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-css.e-
btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-apply.e-flat.e-
primary:disabled {
background-color: brown;
border-color: black;
}
`css
```

Customizing the DateRangePicker popup calendar cancel button in footer

Use the following CSS to customize the DateRangePicker popup calendar cancel button in footer

```
`css
/ To specify background color, color, and border color /
.e-daterangepicker.e-popup .e-footer .e-btn.e-flat, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-flat {
background-color: beige;
border-color: black;
color: maroon;
}
`
```

Customizing the footer element in the DateRangePicker popup calendar

Use the following CSS to customize the DateRangePicker popup calendar footer element

```
`css
/ To specify background color, color, and border color /
.e-daterangepicker.e-popup .e-footer {
background-color: beige;
height: 50px;
}
`
```

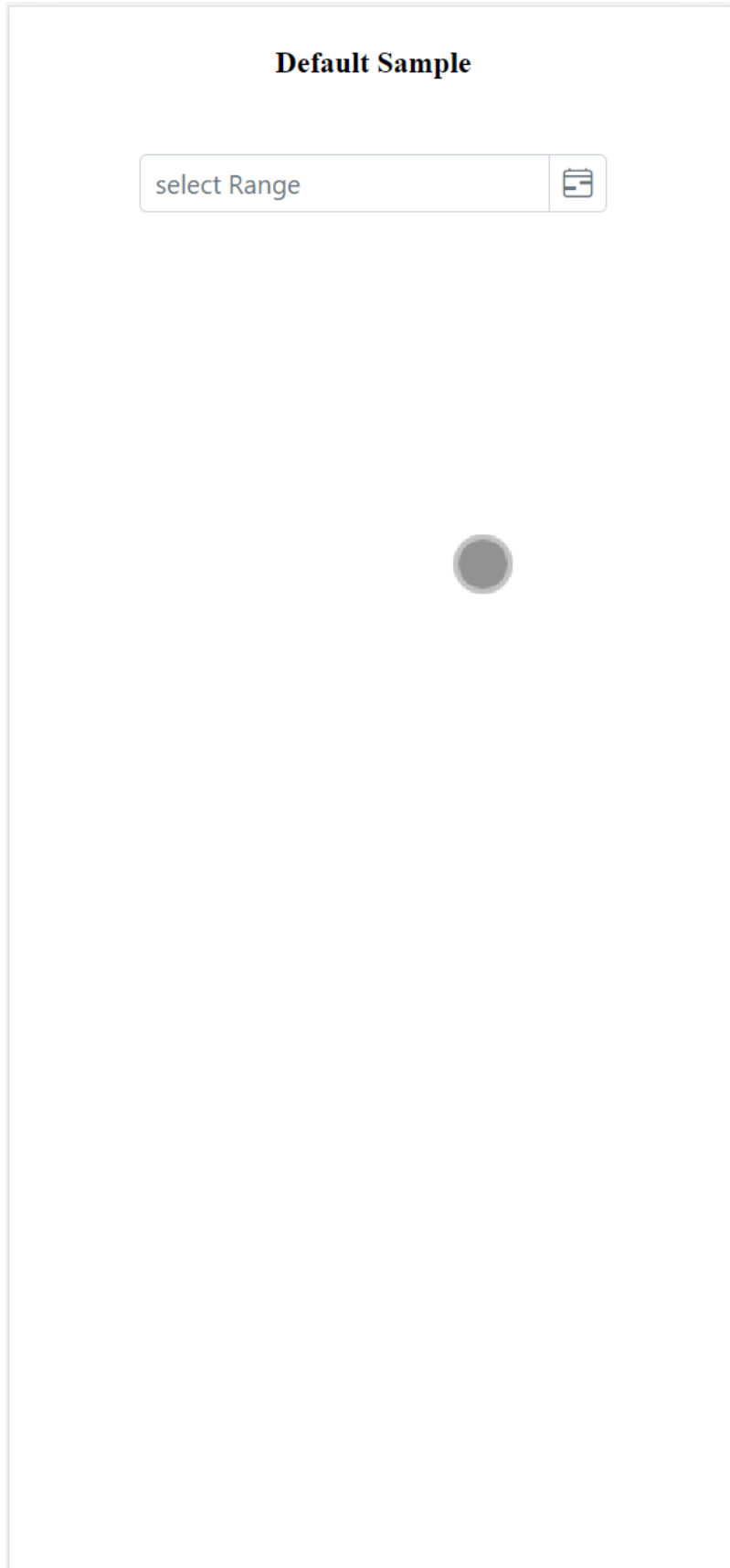
Customizing the selected date cell grid in the DateRangePicker popup calendar

Use the following CSS to customize the selected date cell grid in the DateRangePicker popup calendar

```
`css
/ To specify background and border /
.e-calendar .e-content td.e-focused-date.e-today span.e-day {
background: lightgrey;
border: 1px solid black;
}
`
```

Full screen mode support in mobiles and tablets

The DateRangePicker full-screen mode feature enables users to view the popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DateRangePicker, simply set the [FullScreenMode](#) API value to `true`. This action will extend the calendar and presets popup element to occupy the entire screen on mobile devices.



![@Html.PickersFullScreen](./images/DateRangePickerPresetsFullScreen.gif)

How To

Disable the control

DateRangePicker can be inactivated on a page, by setting [enabled](#) value as false that will disable the control completely from all the user interactions including in the form post.

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").Enabled("false").Render()
```

DISABLE.CS

Set the placeholder

The following example demonstrates how to set [placeholder](#) in the DateRangePicker control.

Using **placeholder** you can display a short hint in the input element.

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").Placeholder("Choose a range").Render()
```

PLACEHOLDER.CS

Customization using cssClass

To customize UI, you can make use of [cssClass](#) that will be added to the DateRangePicker control as the root CSS class. With this CSS class, you can override existing styles of DateRangePicker.

Following is the list of classes that provides flexible way to customize the DateRangePicker control.

Class Name	Description
---	---
e-date-range-wrapper	Applied to DateRangePicker wrapper.
e-range-icon	Applied to DateRangePicker icon.
e-popup	Applied to DateRangePicker popup wrapper.
e-calendar	Applied to both Calendar element.
e-right-calendar	Applied to right Calendar element.
e-left-calendar	Applied to left Calendar element.
e-start-label	Applied to start label in a popup.
e-end-calendar	Applied to end label in a popup.
e-day-span	Applied to day span details label in a popup.

- | e-footer | Applied to footer elements in a popup. |
- | e-apply | Applied to apply button in footer in a popup. |
- | e-cancel | Applied to cancel button in footer in a popup. |
- | e-header | Applied to Calendar header. |
- | e-title | Applied to Calendar title. |
- | e-icon-container | Applied to Calendar previous and next icon container. |
- | e-prev | Applied to Calendar previous icon. |
- | e-next | Applied to Calendar next icon. |
- | e-weekend | Applied to Calendar weekend dates. |
- | e-other-month | Applied to Calendar other month dates. |
- | e-day | Applied to each day cell of the Calendar. |
- | e-selected | Applied to Calendar selected dates. |
- | e-disabled | Applied to Calendar disabled dates. |

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").CssClass("customCSS").Placeholder("Select Range").Render()

<style>
    .customCSS .e-calendar .e-content .e-selected span.e-day, /* csslint
allow: adjoining-classes*/
    .customCSS .e-calendar .e-content .e-selected span.e-day:hover, /*
csslint allow: adjoining-classes*/
    .customCSS .e-calendar .e-content .e-today.e-selected:hover span.e-day,
/* csslint allow: adjoining-classes*/
    .customCSS .e-calendar .e-content .e-today.e-selected span.e-day, /*
csslint allow: adjoining-classes*/
    .customCSS .e-calendar .e-content .e-selected:hover span.e-day /*
csslint allow: adjoining-classes*/ {
        background-color: #35b86b;
    }
    .customCSS .e-calendar .e-content .e-today span.e-day, /* csslint allow:
adjoining-classes*/
    .customCSS .e-calendar .e-content .e-focused-date.e-today span.e-day {
/* csslint allow: adjoining-classes*/
        border: 1px solid #35b86b;
        color: #ff3337;
    }
    .customCSS .e-calendar .e-content .e-weekend span { /* csslint allow:
adjoining-classes*/
        color: #ff3337;
    }
    .customCSS.e-date-range-wrapper .e-input-group-icon.e-icons.e-active, /*
csslint allow: adjoining-classes*/
    .customCSS .e-btn.e-flat, /* csslint allow: adjoining-classes*/
    .customCSS .e-btn.e-flat:hover { /* csslint allow: adjoining-classes*/
        color: #35b86b;
    }
</style>
```

CSS.CSS

Customize the daterangepicker day header

You can change the format of the day that has to be displayed in header using [dayHeaderFormat](#) property.

Note: By default, the format is **Short**.

You can find the possible formats on below.

| **Name** | **Description** |

| ----- | ----- |

| **Short** | Sets the short format of day name (like Su) in day header. |

| **Narrow** | Sets the single character of day name (like S) in day header. |

| **Abbreviated** | Sets the min format of day name (like Sun) in day header. |

| **Wide** | Sets the long format of day name (like Sunday) in day header. |

CSHTML

```
@Html.EJS().DateRangePicker("daterangepicker").DayHeaderFormat(DayHeaderFormat.Short).Render()
```

HEADERFORMAT.CS

The screenshot shows a date range picker interface. It has two main sections: 'Start Date' and 'End Date', separated by a right-pointing arrow. Below these is a 'Selected Days' section. The 'Start Date' calendar shows July 2019, with the 2nd highlighted in a red circle. The 'End Date' calendar shows August 2019. At the bottom, there are 'CANCEL' and 'APPLY' buttons.

This screenshot is identical to the one above, showing the DateRangePicker component with the 2nd of July 2019 selected in the Start Date calendar.

Render DateRangePickerFor

The `DateRangePickerFor` component can be rendered by passing value from the model. The selected date range value can be retrieved during form submission using the post method at the server end.

CSHTML

```
@model EJ2CoreSampleBrowser.Controllers.DateRangePicker
@using (Html.BeginForm())
{
    @Html.EJS().DateRangePickerFor(model => model.value).Render()
    <div>
        @Html.ValidationMessageFor(model => model.value)
    </div>
    <div id="submitButton">
```

```
@Html.EJS().Button("btn").Content("Post").Render()  
</div>  
}
```

DATERANGEPICKERFOR.CS

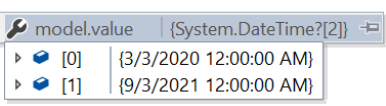
```
using System;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
namespace EJ2CoreSampleBrowser.Controllers  
{  
    public class DateRangePicker  
    {  
        [Required(ErrorMessage = "Please enter the value")]  
        public DateTime?[] value { get; set; }  
    }  
    public class HomeController : Controller  
    {  
        DateRangePicker DateRangeValue = new DateRangePicker();  
        public ActionResult Index()  
        {  
            DateRangeValue.value = new DateTime?[] { new DateTime(2020, 03,  
03), new DateTime(2021, 09, 03) };  
            return View(DateRangeValue);  
        }  
        [HttpPost]  
        public ActionResult Index(DateRangePicker model)  
        {  
            //posted value is obtained from the model  
            DateRangeValue.value = model.value;  
            var startDate = model.value[0];  
            var endDate = model.value[1];  
            return View(DateRangeValue);  
        }  
    }  
}
```

```

public class DateRangePicker
{
    [Required(ErrorMessage = "Please enter the value")]
    public DateTime?[] value { get; set; }
}

0 references
public class HomeController : Controller
{
    DateRangePicker DateRangeValue = new DateRangePicker();
    public ActionResult Index()
    {
        DateRangeValue.value = new DateTime?[] { new DateTime(2020, 03, 03), new DateTime(2021, 09, 03) };
        return View(DateRangeValue);
    }
    [HttpPost]
    public ActionResult Index(DateRangePicker model)
    {
        //posted value is obtained from the model
        DateRangeValue.value = model.value;
        var startDate = model.value[0];
        var endDate = model.value[1];
        return View(DateRangeValue);
    }
}

```



DateTimePicker

Getting Started with ASP.NET MVC DateTimePicker Control

This section briefly explains about how to include [ASP.NET MVC DateTimePicker](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

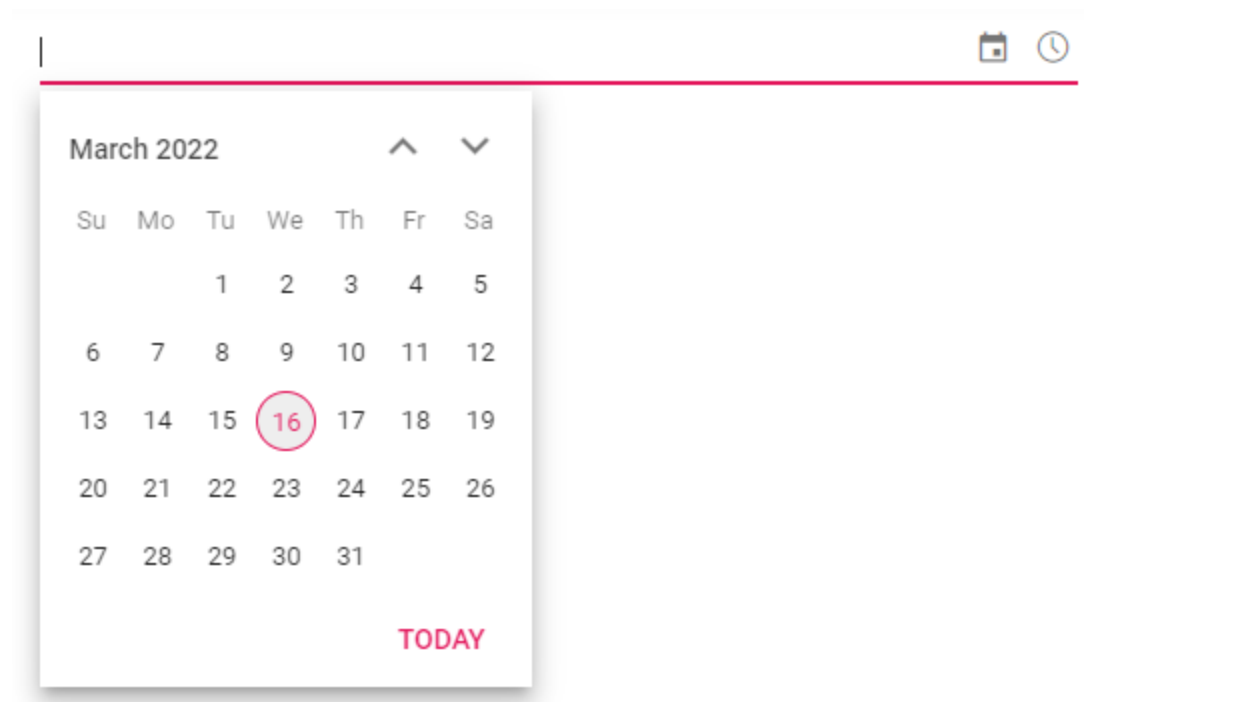
Add ASP.NET MVC DateTimePicker control

Now, add the Syncfusion ASP.NET MVC DateTimePicker control in **~/Views/Home/Index.cshtml** page.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DateTimePicker control will be rendered in the default web browser.

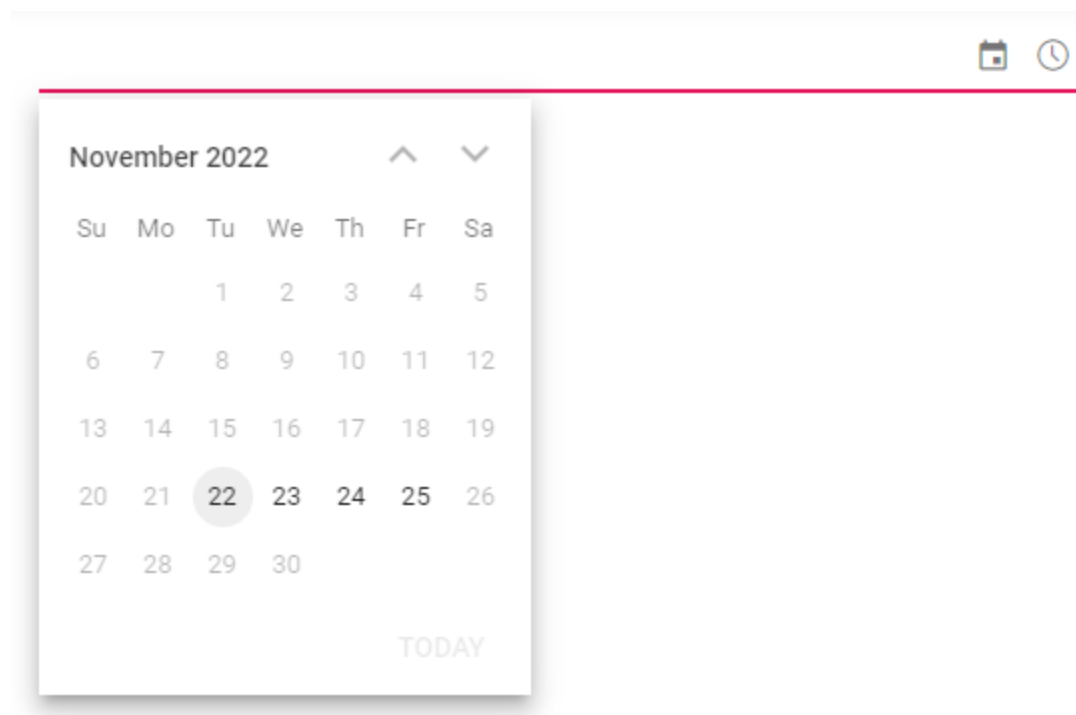


Setting the min and max

The minimum and maximum date time can be defined with the help of [Min](#) and [Max](#) property. The following example demonstrates to set the `min` and `max` on initializing the DateTimePicker.

CSHTML

```
@{
    .....
    var minDate = new DateTime(2022, 11, 22, 12, 00, 00);
    var maxDate = new DateTime(2022, 11, 25, 05, 00, 00);
}
@Html.EJS().DateTimePicker("datetimepicker").Max(maxDate).Min(minDate).Render()
```



Note: [View Sample in GitHub.](#)

See also

- [Render DateTimePicker with specific culture](#)
- [How to get and set value in DateTimePickerFor](#)

Strict Mode

The [strictMode](#) is an act, that allows the user to enter only the valid date and time within the specified min/max range in textbox. If the input entered is invalid, then the control will stay with the previous value. Else, if the datetime is out of range, then the control will set the datetime to the min/max value.

The following example demonstrates the DateTimePicker in `strictMode` with min/max range of 5/5/2019 2:00 AM to 5/25/2019 2:00 AM. Here, it allows to enter only the valid date and time within the specified range. If you are trying to enter the out-of-range value as 5/28/2019, then the value will set the `max` value as 5/25/2019 2:00 AM, since the value 28 is greater than `max` value of 25. Or else if you are trying to enter the invalid date, then the value will stay with the previous value.

The following example demonstrates the DateTimePicker with `strictMode true`.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").StrictMode(true).Max(ViewBag.maxDate).Min(ViewBag.minDate).Value(@ViewBag.value).Render()
```

STRICTMODE.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;
```

```

using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult StrictMode()
        {
            ViewBag.minDate= new DateTime(2019, 5, 5, 02,00,00);
            ViewBag.maxDate = new DateTime(2019, 5, 25, 02, 00, 00);
            ViewBag.value = new DateTime(2019, 5, 28, 02, 00, 00);
            return View();
        }
    }
}

```

By default, the DateTimePicker act in strictMode **false** state allows to enter the invalid or out-of-range datetime in textbox.

If the datetime is out-of-range or invalid, then the model value will be set to **out of range** datetime value or **null** respectively with highlighted **error** class to indicate that the datetime is out of range or invalid.

The following example demonstrates the strictMode as **false**. Here, it allows to enter the valid or invalid value in textbox. If you are entering the out-of-range or invalid datetime value, then the model value will be set to **out of range** datetime value or **null** respectively with highlighted **error** class to indicate that the datetime is out of range or invalid.

CSHTML

```

@Html.EJS().DateTimePicker("datetimepicker").Max(ViewBag.maxDate).Min(ViewBag.minDate).Value(@ViewBag.value).Render()

```

STRICTMODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult StrictModeDisabled()
        {
            ViewBag.minDate= new DateTime(2017, 5, 5, 02, 00, 00);
            ViewBag.maxDate = new DateTime(2017, 5, 25, 03, 00, 00);
            ViewBag.value = new DateTime(2017, 5, 25, 04, 00, 00);
            return View();
        }
    }
}

```

Note: If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

DateTime Range

DateTimePicker provides an option to select a date and time value within a specified range by using the [min](#) and [max](#) properties. Always the min value has to be lesser than the max value.

When the min and max properties are configured and the selected datetime value is out-of-range or invalid, then the model value will be set to `out of range` datetime value or `null` respectively with highlighted `error` class to indicate that the datetime is out of range or invalid. The value property depends on the min/max with respect to [strictMode](#) property.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Value("@ViewBag.value").Min("@ViewBag.minDate").Max("@ViewBag.maxDate").Render()
```

DATETIMERANGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult DateTimeRange()
        {
            ViewBag.minDate= new
            DateTime(DateTime.Now.Year, DateTime.Now.Month, 7, 0, 0, 0);
            ViewBag.maxDate = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 27, DateTime.Now.Hour, DateTime.Now.Minute,
            DateTime.Now.Second);
            ViewBag.value = new DateTime(DateTime.Now.Year,
            DateTime.Now.Month, 14, DateTime.Now.Hour, DateTime.Now.Minute,
            DateTime.Now.Second);
            return View();
        }
    }
}
```

Note: If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

DateTime Format

DateTime format is a way of representing the date and time value in different string format in textbox.

By default the DateTimePicker's format is based on the culture. You can also set the own custom format by using the [format](#) property.

Note: Once the date format property has been defined it will be common to all the cultures.

To know more about the date and time format standards, refer to the [Internationalization Date Time Format](#) section.

The following example demonstrates the DateTimePicker with the custom format (yyyy-MM-dd hh:mm).

CSHTML

```
@Html.EJS().DateTimePicker("element").Value(ViewBag.value).Format("yyyy-MM-dd hh:mm").Placeholder("Enter date and time").Render()
```

DATE-TIME-FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
namespace EJ2CoreSampleBrowser.Controllers
{
    public partial class HomeController : Controller
    {
        // GET: /<controller>/
        public IActionResult DateTimeFormat()
        {
            ViewBag.value = new
            DateTime(DateTime.Now.Year, DateTime.Now.Month, 14);
            return View();
        }
    }
}
```

Enable the Masked Input

DateTimePicker has `enableMask` property that provides the option to enable the built-in date masking support.

CSHTML

```
@Html.EJS().DateTimePicker("element").EnableMask("true").Render()
```

MASK-INPUT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Sample()
        {
        }
    }
}
```

```

        return View();
    }
}

```

The mask pattern is defined based on the provided date format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| **Keys** | **Actions** |

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the date and time. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

CSHTML

```

<div class="control-section">
    <div class="control_wrapper">
        <div class="pane">
            <div class="tabs-wrap">
                <div class="wrap">
                    // Specifies the masked DateTimePicker without
format property.
@Html.EJS().DateTimePicker("element").EnableMask("true").Render()
                </div>
            </div>
            <div class="tabs-wrap">
                <div class="wrap">
                    // Specifies the masked DateTimePicker with format.
@Html.EJS().DateTimePicker("element").EnableMask("true").Format("M/d/yyyy").
Render()
                </div>
            </div>
        </div>
    </div>
</div>

```

DATE-FORMAT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Sample()
        {
            return View();
        }
    }
}

```

```
}

```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default, it takes the full name of date and time co-ordinates such as `day`, `month`, `year`, `hour` etc.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of `L10n` class for mask placeholder values like below.

```
`typescript
//load the locale object to set the localized mask placeholder value
L10n.load({
'de': {
'datetimepicker': { day: 'Tag' , month: 'Monat', year: 'Jahr', hour: 'Stunde' ,minute: 'Minute',
second:'Sekunden' }
}
});
`
```

CSHTML

```
<div class="control-section">
    <div class="control_wrapper">
        <div class="pane">
            <div class="tabs-wrap">
                <div class="wrap">
                    // Specifies the masked DateTimePicker with format
property.
@Html.EJS().DateTimePicker("element").EnableMask("true").Render()
                </div>
            </div>
            <div class="tabs-wrap">
                <div class="wrap">
                    // Specifies the masked DateTimePicker without
format.
@Html.EJS().DateTimePicker("element").EnableMask("true").MaskPlaceholder(new
Syncfusion.EJ2.Calendars.DateTimePickerMaskPlaceholder {Day = "d", Month =
"M", Year = "y", Hour="h", Minute="m", Second="s"}).Render()
                </div>
            </div>
        </div>
    </div>
</div>
```

MASK-PLACEHOLDER.CS

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Sample()
        {
            return View();
        }
    }
}
```

Customization

The DateTimePicker is available for UI customization that can be achieved by using available properties and events in the component.

Day and Time Cell format

The DateTimePicker is available for UI customization based on your application requirements. It can be achieved by using [renderDayCell](#) event that provides an option to customize each day cell on rendering.

The following example disables the weekends of every month by using `renderDayCell` event.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Placeholder("Select a date and time").RenderDayCell("onRenderCell").Render()
<script>
    function onRenderCell(args) {
        if (args.date.getDay() == 0 || args.date.getDay() == 6) {
            //sets isDisabled to true to disable the date.
            args.isDisabled = true;
        }
    }
</script>
```

DAYTIMECELLFORMAT.CS

See Also

- [How to disable the DateTimePicker control](#)
- [How to customize the DateTimePicker day header](#)

Globalization in DateTimePicker Control

Globalization is the combination of adapting the control to various languages by means of parsing and formatting the date or number [Internationalization](#) and also by adding cultural specific customizations and translating the text [localization](#).

By default, the date format, week,time format ,month and meridian names are specific to the **American English** culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the **loadCldr** method to load culture specific CLDR JSON data.

- Set the culture by using the [locale](#) property.

To go with the different culture other than **English**, follow the below steps.

- Install the **CLDR-Data** package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the [CLDR-Data](#) link.

`

```
npm install cldr-data --save
```

`

Once the package installed, you can find the culture specific JSON data under the location **/node_modules/cldr-data**.

In ASP.NET MVC refer the culture files directly from **/node_modules/cldr-data** location.

```
`typescript
```

```
function loadCultureFiles(name) {  
    var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];  
    var loader = ej.base.loadCldr;  
    var loadCulture = function (prop) {  
        var val, ajax;  
        ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-data/main/' +  
            name + '/' + files[prop], 'GET', false);  
        ajax.onSuccess = function (value) {  
            val = value;  
        };  
        ajax.send();  
        loader(JSON.parse(val));  
    };  
    for (var prop = 0; prop < files.length; prop++) {  
        loadCulture(prop);  
    }  
}
```

`

Note: The **Localization** library allows you to localize default text content of the DateTimePicker. The DateTimePicker control has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the **locale** value and translation object.

| Locale keywords | Text |

| ----- | ----- |

| today | Name of the button to choose Today date. |

| placeholder | Hint to describe expected value in input element. |

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of **L10n** class.

`typescript

```
var L10n = ej.base.L10n;
```

```
L10n.load({
```

```
'de': {
```

```
'datetimepicker': { placeholder: "Wählen Sie ein Datum und eine Uhrzeit aus",
```

```
today:"heute"
```

```
}
```

```
}
```

```
});
```

```
`
```

The following example demonstrates the DateTimePicker in **German** culture.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        datetimepicker =
document.getElementById('datetimepicker').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            'de': {
                'datetimepicker': {
                    placeholder: "Wählen Sie ein Datum und eine Uhrzeit
aus",
                    today: "heute"
                }
            }
        });
        loadCultureFiles('de');
        datetimepicker.locale = 'de';
    });
    function loadCultureFiles(name) {
```

```

        var files = ['ca-gregorian.json', 'numbers.json',
        'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../.../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
            } else {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../.../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
        for (var prop = 0; prop < files.length; prop++) {
            loadCulture(prop);
        }
    }
</script>

```

GLOBALIZATION.CS

Right-To-Left

The DateTimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction.

The following code example initialize the DateTimePicker control in Arabic culture and also explains how to set the localized text to the placeholder using load method of L10n class.

The following example demonstrates DateTimePicker in Arabic culture with right-to-left direction.

CSHTML

```

@Html.EJS().DateTimePicker("datetimepicker").EnableRtl(true).Render()
<script>
    document.addEventListener('DOMContentLoaded', function () {
        datetimepicker =
document.getElementById('datetimepicker').ej2_instances[0];
        var L10n = ej.base.L10n;
        L10n.load({
            'ar': {
                'datetimepicker': {
                    placeholder: 'حدد التاريخ والوقت',
                    today: 'اليوم'
                }
            }
        });
    });
</script>

```

```

    }
  }
});
loadCultureFiles("ar");
datetimepicker.locale = 'ar';
});
function loadCultureFiles(name) {
  var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
  if (name === 'ar') {
    files.push('numberingSystems.json');
  }
  var loader = ej.base.loadCldr;
  var loadCulture = function (prop) {
    var val, ajax;
    if (name === 'ar' && prop === files.length - 1) {
      ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
    } else {
      ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
    }
    ajax.onSuccess = function (value) {
      val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
  };
  for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}
</script>

```

RTL.CS

Accessibility

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

The DateTimePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DateTimePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |


```

| WCAG 2.2 Support |  |
| Section 508 Support |  |
| Screen Reader Support |  |
| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

DateTimePicker provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA support is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied to the input element.

To know about the accessibility of Calendar, refer to the Calendar's [Accessibility](#) section.

It helps to provide information about the widget for assistive technology to the disabled person in screen reader.

- **Aria-expanded:** attributes indicates the state of a collapsible element.

- **Aria-disabled:** attribute indicates the disabled state of this DateTimePicker control.
- **Aria-activedescendent:** attribute helps in managing the current active child of the DateTimePicker control.

Keyboard Interaction

You can use the following keys to interact with the DateTimePicker. The control implements the keyboard navigation support by following the [WAI-ARIA practices](#).

DateTimePicker supports the below list of shortcut keys.

Input Navigation

Before opening the popup, use the below list of keys to `DateTimePicker` control the popup element.

| **Press** | **To do this** |

| --- | --- |

| **Alt + Down Arrow** | **Open the select popup** |

| **Alt + Down Arrow + Alt + Down Arrow** | **Toggle between two popups** |

Calendar Navigation

Use the below list of keys to interact with the Calendar after the DatePicker popup has opened.

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | **Focus the previous week date.** |

| **Down Arrow** | **Focus the next week date.** |

| **Left Arrow** | **Focus the previous date.** |

| **Right Arrow** | **Focus the next date.** |

| **Home** | **Focus the first date in the month.** |

| **End** | **Focus the last date in the month.** |

| **Page Up** | **Focus the same date in the previous month.** |

| **Page Down** | **Focus the same date in the next month.** |

| **Enter** | **Select the currently focused date.** |

| **Shift + Page Up** | **Focus the same date in the previous year.** |

| **Shift + Page Down** | **Focus the same date in the previous year.** |

| **Control + Upper Arrow** | **Moves into the inner level of view like month-year, year-decade.** |

| **Control + Down Arrow** | **Moves out from the depth level view like decade-year, year-month.** |

| **Control +Home** | **Focus the starting date in the current year.** |

| **Control +End** | **Focus the ending date in the current year.** |

Use the below list of shortcut keys to interact with the TimePicker after the TimePicker Popup has opened.

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | **Navigate and select the previous item.** |

| **Down Arrow** | **Navigate and select the next item.** |

| **Left Arrow** | **Move the cursor towards arrow key pressed direction.** |

| **Right Arrow** | **Move the cursor towards arrow key pressed direction.** |

| **Home** | **Navigate and select the first item.** |

| **End** | **Navigate and select the last item.** |

| **Enter** | **Select the currently focused item and close the popup.** |

| **Alt + Upper Arrow** | **Close the popup.** |

| **Alt + Down Arrow** | **Open the popup.** |

| **Esc** | **Close the popup.** |

Note: To focus the DateTimePicker control use the **alt+t** keys.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Placeholder("Select a date and time").Render()
<script>
    document.onkeyup = function (e) {
        if (e.altKey && e.keyCode === 84 /* t */) {
            // press alt+t to focus the component.
        }
    }
    document.getElementById("datetimepicker").ej2_instances[0].element.focus();
}
</script>
```

ACCESSIBILITY.CS

Ensuring accessibility

The DateTimePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DateTimePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DateTimePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Style and appearance in DateTimePicker Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DateTimePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height and font size /
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {
font-size: 20px;
height: 40px;
}
`
```

Customizing the DateTimePicker icons element

Use the following CSS to customize the DateTimePicker icons element.

```
`css
/ To specify background color and font size /
.e-datetime-wrapper .e-input-group-icon.e-date-icon, .e-datetime-wrapper .e-input-group-icon.e-time-
icon {
font-size: 16px;
background-color: blanchedalmond;
}
`
```

Customizing the time picker popup in the DateTimePicker

Use the following CSS to customize the time picker popup in the DateTimePicker.

```
`css
/ To specify height /
.e-datetimepicker.e-popup {
height: 100px;
}
`
```

Customizing the Calendar popup of the DateTimePicker



Check the below section, to customize the style and appearance of the Calendar component in the DateTimePicker.

[Customizing Calendar's style and appearance](#)

Full screen mode support in mobiles and tablets

The DateTimePicker full-screen mode feature enables users to view the popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DateTimePicker, simply set the [FullScreenMode](#) API value to `true`. This action will extend the calendar and time popup element to occupy the entire screen on mobile devices.

Default Sample

12/15/2017 2:00 PM		
--------------------	---	---

How To

Disable the component

To disable the DateTimePicker, use its [enable](#) property to `false`.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Enabled(false).Render()
```

DISABLE.CS

Customize the datetimepicker day header

You can change the format of the day that has to be displayed in header using [dayHeaderFormat](#) property.

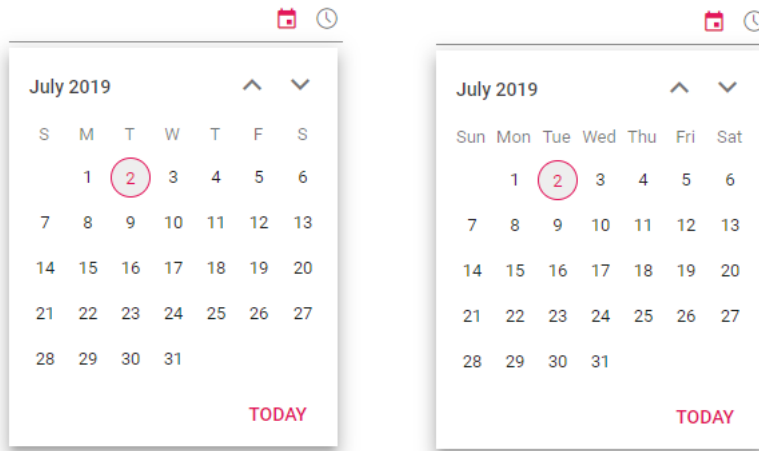
Note: By default, the format is `Short`.

Name	Description
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").DayHeaderFormat(DayHeaderFormat.Short).Render()
```

HEADERFORMAT.CS



Set the placeholder

The following example demonstrates how to set `placeholder` in the DateTimePicker control. Using [placeholder](#) you can display a short hint in the input element.

CSHTML

```
@Html.EJS().DateTimePicker("datetimepicker").Placeholder("Enter
datetime").Render()
```

PLACEHOLDER.CS

Render DateTimePickerFor

The DateTimePickerFor component can be rendered by passing a value from the model. The selected date value can be retrieved during form submission using the post method at the server end.

CSHTML

```
@model EJ2CoreSampleBrowser.Controllers.DateTimePicker
@using (Html.BeginForm())
{
    @Html.EJS().DateTimePickerFor(model => model.value).Render()
    <div>
        @Html.ValidationMessageFor(model => model.value)
    </div>
    <div id="submitButton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}
```

DATETIMEPICKERFOR.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace EJ2CoreSampleBrowser.Controllers
```



```


{
    public class DateTimePicker
    {
        [Required(ErrorMessage = "Please enter the value")]
        public DateTime? value { get; set; }
    }
    public class HomeController : Controller
    {
        DateTimePicker DateTimePickerValue = new DateTimePicker();
        public ActionResult Index()
        {
            DateTimePickerValue.value = new DateTime(2020, 03, 03, 10, 00,
00);
            return View(DateTimePickerValue);
        }
        [HttpPost]
        public ActionResult Index(DateTimePicker model)
        {
            //posted value is obtained from the model
            DateTimePickerValue.value = model.value;
            return View(DateTimePickerValue);
        }
    }
}

```

```

public class HomeController : Controller
{
    DateTimePicker DateTimePickerValue = new DateTimePicker();
    0 references
    public ActionResult Index()
    {
        DateTimePickerValue.value = new DateTime(2020, 03, 03, 10, 00, 00);
        return View(DateTimePickerValue);
    }
    [HttpPost]
    0 references
    public ActionResult Index(DateTimePicker model)
    {
        //posted value is obtained from the model
        ▶ DateTimePickerValue.value = model.value;
        return View(DateTimePickerValue);
    }
}

```

▶  model.value | {3/3/2020 10:00:00 AM} ➡

Diagram

Getting Started with ASP.NET MVC Diagram Control

This section briefly explains about how to include [ASP.NET MVC Diagram](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC

controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Diagram control

Now, add the Syncfusion ASP.NET MVC Diagram control in `~/Views/Home/Index.cshtml` page. You can create and add a **node** (JSON data) with specific position, size, label, and shape.

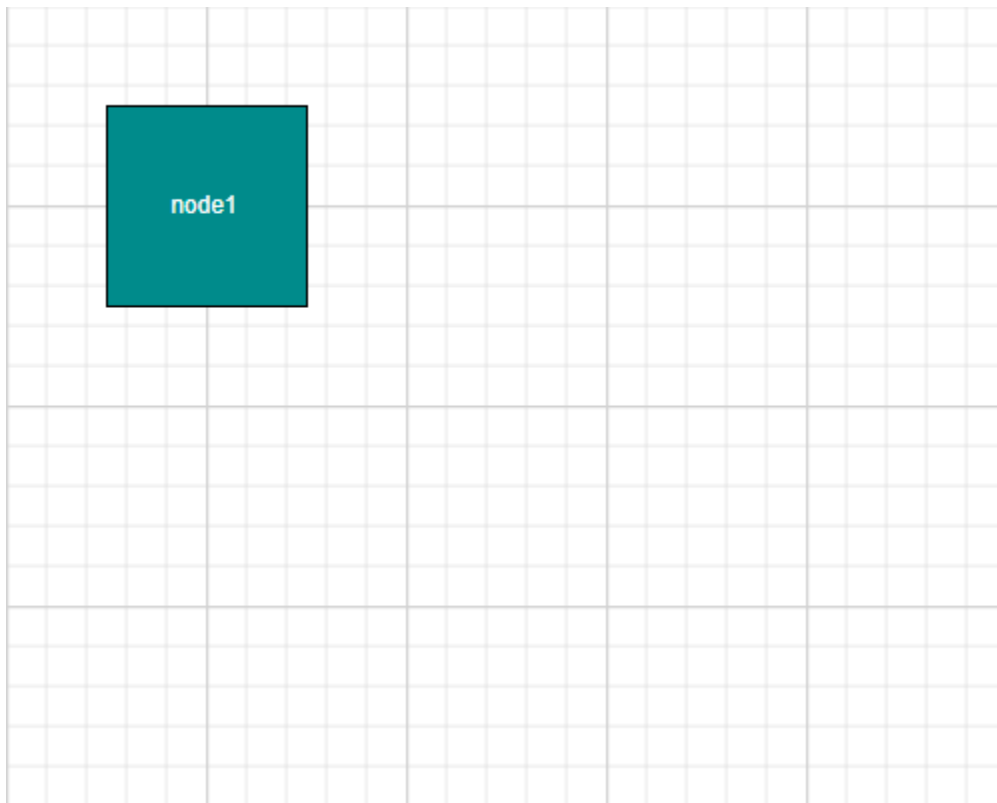
CSHTML

```
@(Html.EJS().Diagram("container")
    .Width("100%")
    .Height("700px")
    .Nodes(ViewBag.nodes).Render())
```

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        // GET: Nodes
        public ActionResult Nodes()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
            List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "node1", Style
            = new DiagramTextStyle() { Color = "White", StrokeColor = "None" } });
            nodes.Add(new Node()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() { Fill = "darkcyan" },
                text = "node1",
                OffsetX = 100,
```

```
        OffsetY = 100,  
        Annotations = Node1  
    });  
    ViewBag.nodes = nodes;  
    return View();  
}  
}  
  
public class Node : DiagramNode  
{  
    public string text;  
}  
}
```



Connect two Nodes with a Connector

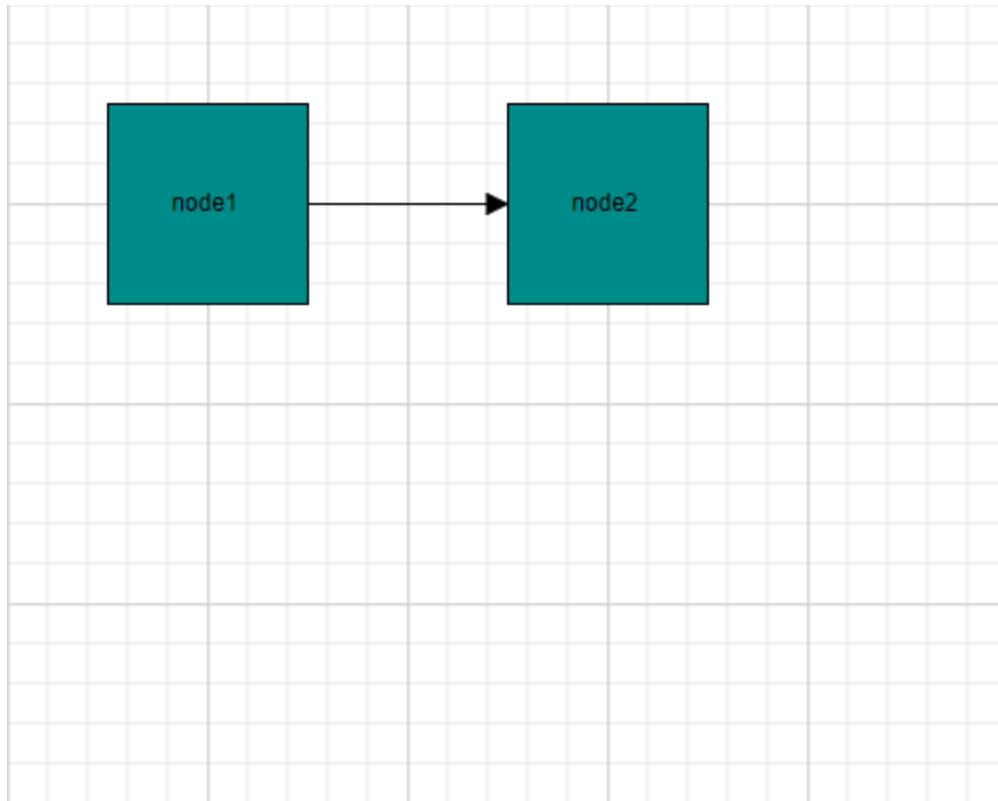
Add two node to the diagram as shown in the previous example. Connect these nodes by adding a connector using the `connector` property and refer the source and target end by using the `sourceNode` and `targetNode` properties.

CSHTML

```
@(Html.EJS().Diagram("container")  
    .Width("100%")  
    .Height("580px")  
    .Nodes(ViewBag.nodes)  
    .Connectors(ViewBag.connectors).Render())
```

CONNECTOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace EJ2MVCSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        // GET: Connector
        public ActionResult Connector()
        {
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "node1" });
            List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
            Node2.Add(new DiagramNodeAnnotation() { Content = "node2" });
            List<DiagramNodeAnnotation> Node3 = new
List<DiagramNodeAnnotation>();
            Nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Annotations = Node1,
                Style = new NodeStyleNodes() { Fill = "darkcyan" },
                OffsetX= 100,
                OffsetY= 100,
                Width = 100,
                Height = 100
            });
            Nodes.Add(new DiagramNode()
            {
                Id = "node2",
                Annotations = Node2,
                Style = new NodeStyleNodes() { Fill = "darkcyan" },
                OffsetX = 300,
                OffsetY = 100,
                Width = 100,
                Height = 100
            });
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID = "node1", TargetID = "node2", });
            ViewBag.nodes = Nodes;
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}
```



Note: [View Sample in GitHub.](#)

Adding default values

Default values for all nodes and connectors can be set using the `getNodeDefaults` and `getConnectorDefaults` properties, respectively. For example, if all nodes have the same width and height, such properties can be moved into `getNodeDefaults`.

For example, if all the nodes have same height and width, we can set such properties as follows.

CSHTML

```
@(Html.EJS().Diagram("container")
    .Width("100%")
    .Height("580px")
    .getNodeDefaults("getNodeDefaults")
    .getConnectorDefaults("getConnectorDefaults")
    .Nodes(ViewBag.nodes)
    .Connectors(ViewBag.connectors).Render()
    function getNodeDefaults(obj, diagram) {
        obj.width = 100;
        obj.height = 100;
        obj.OffsetX = 300;
    }
    function getConnectorDefaults(obj, diagram) {
        obj.type = 'Orthogonal';
    }
)
```

DEFAULTVALUES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace EJ2MVCSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        public ActionResult Connector()
        {
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "node1" });
            List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
            Node2.Add(new DiagramNodeAnnotation() { Content = "node2" });
            List<DiagramNodeAnnotation> Node3 = new
List<DiagramNodeAnnotation>();
            Nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Annotations = Node1,
                Style = new NodeStyleNodes() { Fill = "darkcyan" },
                OffsetY= 100,
            });
            Nodes.Add(new DiagramNode()
            {
                Id = "node2",
                Annotations = Node2,
                Style = new NodeStyleNodes() { Fill = "darkcyan" },
                OffsetY = 300,
            });
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID = "node1", TargetID = "node2", });
            ViewBag.nodes = Nodes;
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

Flow Diagram

Similarly, the required nodes and connectors can be added to form a complete flow diagram.

CSHTML

```

@(Html.EJS().Diagram("container")
    .Width("100%")
    .Height("580px")
    .GetNodeDefaults("getNodeDefaults")
    .GetConnectorDefaults("getConnectorDefaults"))

```

```

        .Nodes (ViewBag.nodes)
        .Connectors (ViewBag.connectors) .Render ()
function getNodeDefaults(obj, diagram) {
    obj.width = 100;
    obj.height = 60;
    obj.offsetX = 300;
}
function getConnectorDefaults(obj, diagram) {
    obj.type = 'Orthogonal';
}

```

FLOWCHART.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace EJ2MVCSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        public ActionResult Connector()
        {
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "node1" });
            List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
            Node2.Add(new DiagramNodeAnnotation() { Content = "node2" });
            List<DiagramNodeAnnotation> Node3 = new
List<DiagramNodeAnnotation>();
            Node3.Add(new DiagramNodeAnnotation() { Content = "i < 10?" });
            List<DiagramNodeAnnotation> Node4 = new
List<DiagramNodeAnnotation>();
            Node4.Add(new DiagramNodeAnnotation() { Content =
"print(hello!!)", Style = new DiagramTextStyle() { Fill = "White" } });
            List<DiagramNodeAnnotation> Node5 = new
List<DiagramNodeAnnotation>();
            Node5.Add(new DiagramNodeAnnotation() { Content = "i++;"});
            List<DiagramNodeAnnotation> Node6 = new
List<DiagramNodeAnnotation>();
            Node6.Add(new DiagramNodeAnnotation() { Content = "End" });
            List<DiagramConnectorAnnotation> connector1 = new
List<DiagramConnectorAnnotation>();
            connector1.Add(new DiagramConnectorAnnotation() { Content =
"Yes" });
            List<DiagramConnectorAnnotation> connector2 = new
List<DiagramConnectorAnnotation>();
            connector2.Add(new DiagramConnectorAnnotation() { Content = "No"
});
            Nodes.Add(new DiagramNode()
            {
                Id = "node1",

```



```

        Annotations = Node1,
        Style = new NodeStyleNodes() { Fill = "darkcyan" },
        Shape = new { type = "Flow", shape = "Terminator" },
        OffsetY = 50,
    });
    Nodes.Add(new DiagramNode()
    {
        Id = "node2",
        Annotations = Node2,
        Style = new NodeStyleNodes() { Fill = "darkcyan" },
        OffsetY = 140,
        Shape = new { type = "Flow", shape = "Process" },
    });
    Nodes.Add(new DiagramNode()
    {
        Id = "node3",
        Annotations = Node3,
        Style = new NodeStyleNodes() { Fill = "darkcyan" },
        OffsetY = 230,
        Shape = new { type = "Flow", shape = "Decision" },
    });
    Nodes.Add(new DiagramNode()
    {
        Id = "node4",
        Annotations = Node4,
        Style = new NodeStyleNodes() { Fill = "darkcyan" },
        OffsetY = 320,
        Shape = new { type = "Flow", shape = "PreDefinedProcess" },
    });
    Nodes.Add(new DiagramNode()
    {
        Id = "node5",
        Annotations = Node5,
        Style = new NodeStyleNodes() { Fill = "darkcyan" },
        OffsetY = 410,
        Shape = new { type = "Flow", shape = "Process" },
    });
    Nodes.Add(new DiagramNode()
    {
        Id = "node6",
        Annotations = Node6,
        Style = new NodeStyleNodes() { Fill = "darkcyan" },

        OffsetY = 500,
        Shape = new { type = "Flow", shape = "Terminator" },
    });
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() { Id = "connector1",
SourceID = "node1", TargetID = "node2", });
    Connectors.Add(new DiagramConnector() { Id = "connector2",
SourceID = "node2", TargetID = "node3", });
    Connectors.Add(new DiagramConnector() { Id = "connector3",
SourceID = "node3", TargetID = "node4", Annotations = connector1, });
    Connectors.Add(new DiagramConnector() { Id = "connector4",
SourceID = "node3", TargetID = "node6", Annotations = connector2, });

```

```

        Connectors.Add(new DiagramConnector() { Id = "connector5",
SourceID = "node4", TargetID = "node5" });
        Connectors.Add(new DiagramConnector() { Id = "connector6",
SourceID = "node5", TargetID = "node3" });
        ViewBag.nodes = Nodes;
        ViewBag.connectors = Connectors;
        return View();
    }
}

```

Automatic Organization Chart

In the 'Flow Diagram' section, how to create a diagram manually was discussed. This section explains how to create and position the diagram automatically.

Business object (Employee information)

Define Employee Information as JSON data. The following code example shows an employee array whose, **Name** is used as an unique identifier and **ReportingPerson** is used to identify the person to whom an employee report to, in the organization.

```

`cs
public data: Object[] = [
{
    Name: "Elizabeth",
    Role: "Director"
},
{
    Name: "Christina",
    ReportingPerson: "Elizabeth",
    Role: "Manager"
},
{
    Name: "Yoshi",
    ReportingPerson: "Christina",
    Role: "Lead"
},
{
    Name: "Philip",
    ReportingPerson: "Christina",
    Role: "Lead"
},

```

```

{
  Name: "Yang",
  ReportingPerson: "Elizabeth",
  Role: "Manager"
},
{
  Name: "Roland",
  ReportingPerson: "Yang",
  Role: "Lead"
},
{
  Name: "Yvonne",
  ReportingPerson: "Yang",
  Role: "Lead"
}
];
`

```

Map data source

You can configure the above "Employee Information" with diagram, so that the nodes and connectors are automatically generated using the mapping properties. The following code example show how `dataSourceSettings` is used to map ID and parent with property name identifiers for employee information.

```

`cs
<ejs-diagram id="diagram" width="100%" height="550px"
getNodeDefaults="@ViewBag.getNodeDefaults"
getConnectorDefaults="@ViewBag.getConnectorDefaults" created="diagramCreated">
<e-diagram-datasourcesettings id="Id" parentId="ReportingPerson" dataManager="new
DataManager(){ Data = (List<OverviewData>)ViewBag.Nodes }"></e-diagram-datasourcesettings>
<e-diagram-layout type="OrganizationalChart"></e-diagram-layout>
</ejs-diagram>
function getNodeDefaults(obj, diagram) {
  obj.height = 60;
  obj.width = 100;
  return obj;
}

```

```
function getConnectorDefaults(connector, diagram) {
connector.type = 'Orthogonal';
return connector;
}

public partial class DiagramController : Controller
{
// GET: Overview
public ActionResult Overview()
{
    ViewBag.Nodes = OverviewData.GetAllRecords();
    return View();
}
}

public class OverviewData
{
    public string Id { get; set; }
    public string Name { get; set; }
    public string Designation { get; set; }
    public string ReportingPerson { get; set; }
    public OverviewData(string id, string name, string designation, string reportingperson)
    {
        this.Id = id;
        this.Name = name;
        this.Designation = designation;
        this.ReportingPerson = reportingperson;
    }
    public static List<OverviewData> GetAllRecords()
    {
        List<OverviewData> data = new List<OverviewData>();
        data.Add(new OverviewData("parent", "Elizabeth", "Director", "" ));
        data.Add(new OverviewData("1", "Christina", "Manager", "parent"));
        data.Add(new OverviewData("2", "Yoshi", "Lead", "1" ));
        data.Add(new OverviewData("3", "Philip", "Lead", "1"));
    }
}
```

```

data.Add(new OverviewData("4", "Yang", "Manager", "parent"));
data.Add(new OverviewData("5", "Roland", "Lead", "4"));
data.Add(new OverviewData("6", "Yvonne", "Lead", "4"));
return data;
}
}
,

```

Visualize employee

The following code examples indicate how to define the default appearance of nodes and connectors. The `setNodeTemplate` is used to update each node based on employee data.

CSHTML

```

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults(
    "getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSettings(s =>
    s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity)).DataSourceSettings(ss =>
    ss.Id("Id").ParentId("ReportingPerson").DataManager(new DataManager() { Data =
    (List<OverviewData>)ViewBag.Nodes })).Layout(l =>
    l.Type(Syncfusion.EJ2.Diagrams.LayoutType.OrganizationalChart).GetLayoutInfo(
    "getLayoutInfo").SnapSettings(se =>
    se.Constraints(Syncfusion.EJ2.Diagrams.SnapConstraints.None)).Created("diagramCreated").Render()
function getNodeDefaults(obj, diagram) {
    obj.height = 60;
    obj.width = 100;
}
function getConnectorDefaults(obj, diagram) {
    obj.type = 'Orthogonal';
}

```

ORGCHART.CS

```

using EJ2MVCSampleBrowser.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EJ2MVCSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        public ActionResult Overview()
        {
            ViewBag.Nodes = OverviewData.GetAllRecords();
            return View();
        }
    }
}

```

```

public class OverviewData
{
    public string Id { get; set; }
    public string Name { get; set; }
    public string Designation { get; set; }
    public string ReportingPerson { get; set; }

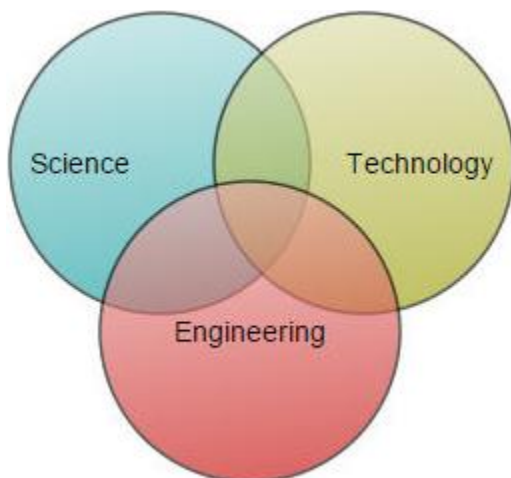
    public OverviewData(string id, string name, string designation,
string reportingperson)
    {
        this.Id = id;
        this.Name =name;
        this.Designation =designation;
        this.ReportingPerson =reportingperson;
    }
    public static List<OverviewData> GetAllRecords()
    {
        List<OverviewData> data = new List<OverviewData>();
        data.Add(new OverviewData("parent", "Elizabeth", "Director", ""
));
        data.Add(new OverviewData("1", "Christina", "Manager",
"parent"));
        data.Add(new OverviewData("2", "Yoshi", "Lead", "1" ));
        data.Add(new OverviewData("3", "Philip", "Lead", "1"));
        data.Add(new OverviewData("4", "Yang", "Manager", "parent"));
        data.Add(new OverviewData("5", "Roland", "Lead", "4"));
        data.Add(new OverviewData("6", "Yvonne", "Lead", "4"));
        return data;
    }
}

```

Note: You can refer to our [ASP.NET MVC Diagram](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Diagram example](#) that shows how to render the Diagram in ASP.NET MVC.

Node in Diagram Control

Nodes are graphical objects used to visually represent the geometrical information, process flow, internal business procedure, entity, or any other kind of data.



<!-- markdownlint-disable MD033 -->

Create node

A node can be created and added to the diagram, either programmatically or interactively. Nodes are stacked on the diagram area from bottom to top in the order they are added.

Add node through nodes collection

To create a node, define the [node](#) object and add that to nodes collection of the diagram model.

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                BorderWidth=2,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Nodel
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

Add/Remove node at runtime

- Nodes can be added at runtime by using public method, add and can be removed at runtime by using public method, remove. On adding node at runtime, the nodes collection is changed and the [collectionChange](#) event will trigger.
- The node's ID property is used to define the name of the node and its further used to find the node at runtime and do any customization.

RUN.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                BorderWidth=2,
                Style = new NodeStyleNodes() {
                    Fill = "Darkcyan",
                    StrokeWidth = 2,
                    StrokeColor = "Black"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Nodel
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

`javascript


```
var node = {
  id: 'node1', offsetX = 100, offsetY = 100, height = 50, width = 100
};
diagram.add(node);
`
```

Add node from palette

Nodes can be predefined and added to the palette, and can be dropped into the diagram when needed. For more information about adding nodes from symbol palette, refer to [Symbol Palette](#).

- Once you drag a node/connector from the palette to the diagram, the following events can be used to do your customization.
- When a symbol is dragged into diagram from symbol palette, the [dragEnter](#) event gets triggered.
- When a symbol is dragged over diagram, the [dragOver](#) event gets triggered.
- When a symbol is dragged and dropped from symbol palette to diagram area, the [drop](#) event gets triggered.
- When a symbol is dragged outside of the diagram, the [dragLeave](#) event gets triggered.

Create node through data source

Nodes can be generated automatically with the information provided through data source. The default properties for these nodes are fetched from default settings. For more information about data source, refer to Data Binding.

Draw nodes

Nodes can be interactively drawn by clicking and dragging the diagram surface by using [NodeDrawingTool](#). For more information about drawing nodes, refer to Draw Nodes.

Position

- Position of a node is controlled by using its [offsetX](#) and [offsetY](#) properties. By default, these offset properties represent the distance between the origin of the diagram's page and node's center point.
- You may expect this offset values to represent the distance between page origin and node's top-left corner instead of center. The Pivot property helps to solve this problem. Default value of node's [pivot](#) point is (0.5, 0.5), that means center of the node.
- The size of the node can be controlled by using its [width](#) and [height](#) properties.
- Rotation of a node is controlled by using its [rotateAngle](#) property.

| Pivot | Offset |

|-----| -----|

| (0.5,0.5) | offsetX and offsetY values are considered as the node's center point. |

| (0,0) | offsetX and offsetY values are considered as the top-left corner of the node. |

| (1,1) | offsetX and offsetY values are considered as the bottom-right corner of the node. |

POSITION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
                Pivot = new DiagramPoint() { X = 1, Y = 1 }
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Flip

The diagram provides support to flip the node. [flip](#) is performed to give the mirrored image of the original element.

The flip types are as follows:

- HorizontalFlip - **Horizontal** is used to change the element in horizontal direction.
- VerticalFlip - **Vertical** is used to change the element in vertical direction.
- Both - **Both** which involves both vertical and horizontal changes of the element.

FLIP.CS

```

using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                flip = FlipDirection.Horizontal,
                Shape = new { type = "Basic", shape = "RightTriangle"},
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Nodel,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Note: The flip is also applicable for group and BPMN shapes.

Appearance

- The appearance of a node can be customized by changing its [fill](#) color, [borderColor](#), [borderWidth](#), [strokeDashArray](#), [opacity](#), and [shadow](#).
- The [visible](#) property of the node enables or disables the visibility of the node.

APPEAR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

```

```

using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                BorderWidth = 2,
                Style = new NodeStyleNodes() {
                    Fill = "Darkcyan",
                    StrokeWidth = 2,
                    StrokeColor = "Black"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Gradient

The [gradient](#) property of the node allows to define and apply the gradient effect to that node.

The gradient stop property defines the color and a position, where the previous color transition ends and a new color transition starts.

The gradient stop's opacity property defines the transparency level of the region.

There are two types of gradients as follows:

- Linear gradient
- Radial gradient

Linear gradient

- [LinearGradient](#) defines a smooth transition between a set of colors (so-called stops) on a line.

- A linear gradient's x1, y1, x2, y2 properties are used to define the position (relative to the node) of the rectangular region that needs to be painted.

LINEARGRAD.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController: Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();

            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation()
            {
                //Sets the offset for the content
                Content = "Node1",
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Fill = "transparent"
                },
            });
            List<stop> Stop = new List<stop>();
            Stop.Add(new stop() { Color = "white", Offset = 0 });
            Stop.Add(new stop() { Color = "#6BA5D7", Offset = 100 });
            linearGradient grade = new linearGradient()
            {
                x1 = 0,
                y1 = 0,
                x2 = 50,
                y2 = 50,
                type = GradientType.Linear,
                Stops = Stop
            };
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width = 100,
                Style = new NodeStyleNodes() { Gradient= grade },
            });
        }
    }
}

```

```
        // add the Annotation for the Node
        Annotations = Model,
    });

    ViewBag.nodes = Nodes;
    return View();
}
}
public class linearGradient
{
    [DefaultValue(null)]
    [HtmlAttributeName("x1")]
    [JsonProperty("x1")]
    public double x1
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("x2")]
    [JsonProperty("x2")]
    public double x2
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("y1")]
    [JsonProperty("y1")]
    public double y1
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("y2")]
    [JsonProperty("y2")]
    public double y2
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public GradientType type
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("stops")]
    [JsonProperty("stops")]
    public List<stop> Stops
    {
        get;
        set;
    }
}
```

```

    }
}
public class stop
{
    [DefaultValue(null)]
    [HtmlAttributeName("color")]
    [JsonProperty("color")]
    public string Color
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("offset")]
    [JsonProperty("offset")]
    public double Offset
    {
        get;
        set;
    }
}
}

```

Radial gradient

- [RadialGradient](#) defines a smooth transition between stops on a circle.
- A radial gradient's cx, cy, fx, fy properties are used to define the position (relative to the node) of the outermost or the innermost circle of the radial gradient.

RADIALGRAD.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController: Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();

            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation()
            {
                //Sets the offset for the content

```

```

        Content = "Node1",
        Style = new DiagramTextStyle()
        {
            Color = "black",
            Fill = "transparent"
        },
    });
    List<stop> Stop = new List<stop>();
    Stop.Add(new stop() { Color = "white", Offset = 0 });
    Stop.Add(new stop() { Color = "#6BA5D7", Offset = 100 });
    radialGradient grade = new radialGradient()
    {
        cx = 50,
        cy = 50,
        fx = 25,
        fy = 25,
        r=50,
        type = GradientType.Radial,
        Stops = Stop
    };
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        Style = new NodeStyleNodes() { Gradient= grade },
        // add the Annotation for the Node
        Annotations = Node1,
    });

    ViewBag.nodes = Nodes;
    return View();
}
}

public class radialGradient
{
    [DefaultValue(null)]
    [HtmlAttributeName("cx")]
    [JsonProperty("cx")]
    public double cx
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("fx")]
    [JsonProperty("fx")]
    public double fx
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("fy")]
    [JsonProperty("fy")]

```



```
public double fy
{
    get;
    set;
}
[DefaultValue(null)]
[HtmlAttributeName("cy")]
[JsonProperty("cy")]
public double cy
{
    get;
    set;
}
[DefaultValue(null)]
[HtmlAttributeName("type")]
[JsonProperty("type")]
public GradientType type
{
    get;
    set;
}
[DefaultValue(null)]
[HtmlAttributeName("r")]
[JsonProperty("r")]
public double r
{
    get;
    set;
}
[DefaultValue(null)]
[HtmlAttributeName("stops")]
[JsonProperty("stops")]
public List<stop> Stops
{
    get;
    set;
}
}
public class stop
{
    [DefaultValue(null)]
    [HtmlAttributeName("color")]
    [JsonProperty("color")]
    public string Color
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("offset")]
    [JsonProperty("offset")]
    public double Offset
    {
        get;
        set;
    }
}
```

```
}

```

Shadow

Diagram provides support to add [shadow](#) effect to a node that is disabled, by default. It can be enabled with the constraints property of the node.

SHADOW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Nodel,
                Constraints=NodeConstraints.Default |
NodeConstraints.Shadow
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

Customizing shadow

The angle, distance, and opacity of the shadow can be customized with the shadow property of the node.

SHADOW2.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
                Constraints=NodeConstraints.Default |
NodeConstraints.Shadow,
                Shadow= new DiagramShadow() { Angle=50, Opacity=0.9}
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Icon

Diagram provides support to describe the state of the node. i.e., the node is expanded or collapsed state.

Note: Icon can be created only when the node has outEdges.

- To explore the properties of expand and collapse icon, refer to [expandIcon](#) and [collapseIcon](#).
- The expandIcon's and collapseIcon's shape properties allows to define the shape of the icon.

ICON.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
            Node2.Add(new DiagramNodeAnnotation() { Content = "Node2" });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
                ExpandIcon=new DiagramIconShape() {
Shape=IconShapes.ArrowDown, Height=10, Width= 10},
                CollapseIcon=new DiagramIconShape() { Shape=IconShapes.ArrowUp, Height=10,
Width= 10}
            });
            nodes.Add(new Node() { Id = "node2", OffsetX = 100, OffsetY = 300,
Annotations = Node2 });
            ViewBag.nodes = nodes;
            List<DiagramConnector> Connectors = new List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector1",
SourceID = "node1", TargetID = "node2" });
            ViewBag.Connectors = Connectors;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Customizing expand icon

- Set the `borderColor`, `borderWidth`, and background color for an `expandIcon` using `borderColor`, `borderWidth`, and `fill` properties.
- Set a size for an `expandIcon` by using `width` and `height` properties.
- The expand icon can be aligned relative to the node boundaries. It has `margin`, `offset`, `horizontalAlignment`, and `verticalAlignment` settings. It is quite tricky, when all four alignments are used together but gives you more control over alignment.
- The [iconColor](#) property can be used to set the `strokeColor` of the Icon.

Customizing collapse icon

- Set the [borderColor](#), [borderWidth](#), background color for an `collapseIcon` using `borderColor`, `borderWidth`, and [fill](#) properties.
- Set a size for `collapseIcon` by using [width](#) and [height](#) properties.
- Like expand icon, collapse icon also can be aligned relative to the node boundaries. It has `margin`, `offset`, `horizontalAlignment`, and `verticalAlignment` settings. It is quite tricky, when all four alignments are used together but gives you more control over alignment.
- The [iconColor](#) property can be used to set the `strokeColor` of the Icon.

Interaction

Diagram provides support to drag, resize, or rotate the node interactively. For more information about editing a node at runtime, refer to [Edit Nodes](#).

Constraints

The `constraints` property of the node allows to enable or disable certain features. For more information about node constraints, refer to [Node Constraints](#).

Custom properties

The [addInfo](#) property of the node allows to maintain additional information to the node.

Stack order

The `nodes z-order` property specifies the stack order of the node. A node with greater stack order is always in front of a node with a lower stack order.

Data flow

Node has the `InEdges` and `OutEdges` read-only property. In this property, you can find what are all the connectors that are connected to the node, and then you can find these connectors by using the [getObject](#) method in the diagram.

DATAFLOW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
```

```

public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
    Node1.Add(new DiagramNodeAnnotation() {
        Content = "node1", Style = new DiagramTextStyle() {
            Color = "White", StrokeColor = "None"
        }
    });
    List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
    Node2.Add(new DiagramNodeAnnotation() {
        Content = "Node2"
    });
    List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
    Node3.Add(new DiagramNodeAnnotation() {
        Content = "Node3"
    });
    List < DiagramNodeAnnotation > Node4 = new List <
DiagramNodeAnnotation > ();
    Node4.Add(new DiagramNodeAnnotation() {
        Content = "Node4"
    });
    nodes.Add(new Node() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            fill = "#6BA5D7",
            strokeColor = "White"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1,
    });
    nodes.Add(new Node() {
        Id = "node2", OffsetX = 100, OffsetY = 300, Annotations =
Node2
    });
    nodes.Add(new Node() {
        Id = "node3", OffsetX = 100, OffsetY = 300, Annotations =
Node3
    });
    nodes.Add(new Node() {
        Id = "node4", OffsetX = 100, OffsetY = 300, Annotations =
Node4
    });
    ViewBag.nodes = nodes;
    List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
    Connectors.Add(new DiagramConnector() {
        Id = "connector1", SourceID = "node1", TargetID = "node2"
    });
    Connectors.Add(new DiagramConnector() {
        Id = "connector2", SourceID = "node1", TargetID = "node3"
    });
}

```

```

    });
    Connectors.Add(new DiagramConnector() {
        Id = "connector3", SourceID = "node1", TargetID = "node4"
    });
    ViewBag.Connectors = Connectors;
    return View();
}
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

See Also

- [How to add annotations to the node](#)
- [How to add ports to the node](#)
- [How to enable/disable the behavior of the node](#)
- [How to add nodes to the symbol palette](#)
- [How to edit the node visual interface](#)
- [How to create diagram nodes using drawing tools](#)

Shapes in Diagram Control

Diagram provides support to add different kind of nodes. They are as follows:

- Text node
- Image node
- HTML node
- Native node
- Basic shapes
- Flow shapes

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-disable MD010 -->

Text

Texts can be added to the diagram as [text](#) nodes. The shape property of the node allows to set the type of node and for text nodes, it should be set as **text**. In addition, define the content object that is used to define the text to be added and style is used to customize the appearance of that text.

TEXT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {

```

```
// GET: Nodes
public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    nodes.Add(new DiagramNode() {
        Id = "node1",
        Width = 100,
        Height = 100,
        OffsetX = 100,
        OffsetY = 100,
        Shape = new { type = "Text", content= "Text" },
        Style = new { color = "White", strokeColor = "None"}
    });
    ViewBag.nodes = nodes;
    return View();
}
}
```

Image

Diagram allows to add images as [image](#) nodes. The shape property of node allows you to set the type of node and for image nodes, it should be set as **image**. In addition, the source property of shape enables you to set the image source.

IMAGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new { type = "Image",
source="/diagram/images/syncfusion.png" },
                Style = new NodeStyleNodes() { Fill="None"}
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

Base64 Encoded Image Into The Image Node:

IMAGEBASE64.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new { type = "Image",
source="data:image/gif;base64,R0lGODlhPQBEAPeoAJosM//AwO/AwHvYZ/z595kzAP/s7P
+goOXMv8+fhw/v739/f+8PD98fH/8mJl+fn/9ZWb8/PzWlWv///6wWGbImAPgTEMImIN9gUFCEm/
gDALULDN8PAD6atYdCTX9gUNKlj8wZAKUsAOzZz+UMAOsJAP/Z2ccMDA8PD/95eX5NWvsJCOVNQP
tfx/8zM8+QePLl38MGBr8JCP+zs9myn/8GBqwpAP/GxgwJCPny78lzlYlgjAJ8vAP9fX/+MjMUCAN
8zM/9wcM8ZGcATEL+QePdZWf/29uc/P9cmJu9MTDImIN+/r7+/vz8/P8VNQGNugV8AAF9fX8swMN
gTAFldOICAgPNSUnNWSMQ5MBAQEJE3QPIGAM9AQmQGcG9vb6MhJsEdGM8vLx8fH98AANIWAMuQeL
8fABkTEPPQ0OM5OSYdGfL5jo+Pj/+pqcsTE78wMFNGQLYmID4dGPvd3UBAQJmTkP+8vH9QUK+vr8
ZWSHpczJmILdwcLOGcHRQUHxwcK9PT9DQ00/v70w5MLypoG8wKOuwsP/g4P/Q0IcwKEswKMl8aJ
9fX2xjdOtGRs/Pz+Dg4GIImIP8gIH0sKEAwKKmTiKZ8aB/f39Wsl+Lft8dgUE9PT5x5aHBwcP+AgP
+WltdgYMyZfyywz78AAAAAAD//8AAP9mZv//wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAACH5BAEAAKgALAAAAAAAEQAAj/AFEJHEiwoMGDCBMqXMiwoCABww4nEhxoYkUpzJGrM
ixogkfGUNqlNixJEIDB0SqHGmyJSojM1bKZOmYop0gM3Oe2liTISKMOoPy7GnwY9CjIYcSRYm0aV
KSLmE6nfq05QycVLPuhDrxB1CtYJUqNaq2bNWEBj6ZXruyxZyDRtqwnXvkhACDV+euTeJm1Ki7A7
3qNwTfiF+/gA95Gly2CJLDhweHMOUAAuOpLYDEgBxZ4GRTlClfDnpkM+fOqD6DDj1aZpITp0dtGC
Dhr+fVuCu3zlg49ijaokTZt027uG7Gjn2P+hI8+PDPERoUB318bWbFAJ5sUNFCuGRTYUqV/3ogfX
plrWlMc6awJjiAA2fm4ogXjz56aypOoIde4OE5u/F9x199dlXnnGiHZWEYbGpsAEA3QXynHwEFl
iKAgswgJ8LPeiUXGwedCAKABACCN+EAlpYIIYaFlcDhytd51sGAJbo3onOpajiihl092KHGaUXGw
WjUBChjSPiWJuOO/LYIm4vltXfE6J4gCSJEZ7YgRYUNrkji9P55sF/ogxw5ZkSqIDaZBV6aSGYq/
lGZplndkckZ98xoICbTcIJGQAZcNmduC210hs35nCyJ58fgmIKX5RQGOZowxaZwYA+JaoKQswG
ijBV4C6SiTUmpphMspJx9unX4KaimjDv9aaXOEbteBqmuuxgEHOlX6Kqx+yXqqBANsgCtit4FWQA
EkrNbpq7HSOmtwag5w57GrmlJBASEU18ADjUYb3ADTinIttsgSBloJFFa63bduimuqKB1keqwUho
CSK374wbujuOSu4QG6UvxBRydcPksav++Ca6G8A6Pr1x2kVMYHwsVxUALDq/krnrhPSOzXG1lUTI
offqGR7Goi2MAxbv602kEG56I7CSlRsEFKfVYovDJoIRTg7sugNRDGqCJzJgcKE0ywc0ELm6KBCC
Jo8DIPFeCWNGcyqNFE06ToAfV0HBRgxsvLThHn1oddQMrXj5DyAQgjEHSAJMWZs3HPxT/QMbabI
/iBclimLEJKX2EEKomBAUCxRi42VDADxyTYDVogV+wSchqmKxEKCDAYFDFj4OmwbY7bDGdBhtrnT
QYOigeChUmc1K3QTnAUfEgGFgAWt88hKA6aCRIXhxnQ1yg3BCayK44EWdkUQcBBYEQChFxfCB776
aQsG0BilQgQgE8qO26X1h8cEUep8ngRBnOy74E9QgRgEAC8SvOfQkh7FDBDmS43PmGoIiKUUEGkM
EC/PJHgXw0xH74yx/3XnaYRjgMB8obxQW6kL9QYEJ0FIFgByfIL7/IQAlvQwEpnAC7DtLNJCKUoO
/w45c44GwCXiAFB/OXAATQryUxdN4LfFiwgjCNYg+kYMIIEFkCKDs6PKAIJouyGWMS1FSKJOMRB/B
oIxYJIUXFUXNwoIkEKPAgCBZSQHQ1A2EWDfDEUvLyADj5AchSIQW6gu10bE/JG2VnCZGfo4R4d0s
dQoBAHhPjhIB94v/wRoRKQWGRHgrhGSQJxCS+0pCZbEhAAOw==" },
                Style = new NodeStyleNodes() { Fill="None" }
            });
        }
    }
}

```

```

        });
        ViewBag.nodes = nodes;
        return View();
    }
}

```

Note: Deploy your HTML file in the web application and export the diagram (image node) or else the image node will not be exported in the Chrome and Firefox due to security issues.

Link 1: [Draw images on canvas locally using chrome](#)

Link 2: [Local image in canvas in chrome](#)

Image alignment

Stretch and align the image content anywhere but within the node boundary.

The scale property of the node allows to stretch the image as you desired (either to maintain proportion or to stretch). By default, the [scale](#) property of the node is set as **meet**.

SCALE.CS

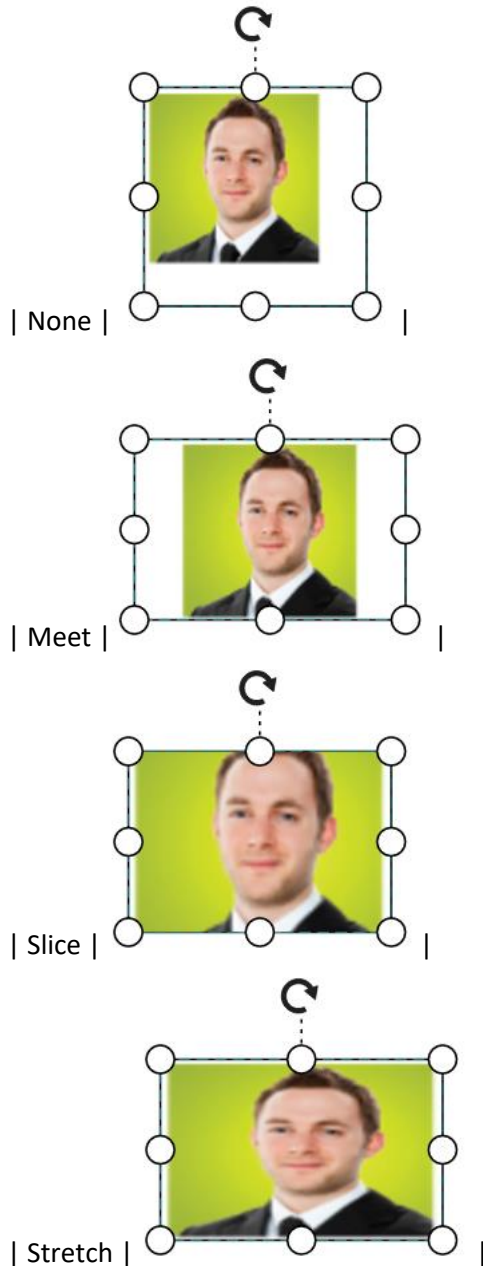
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new { type = "Image",
source="/diagram/images/syncfusion.png", scale="Meet" },
                Style = new NodeStyleNodes() { Fill="None" }
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}

```

| Values | Images |

|-----| -----|



HTML

Html elements can be embedded in the diagram through [Html](#) type node. The shape property of node allows to set the type of node and to create a HTML node it should be set as `HTML`.

HTML.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
```

```

public partial class DiagramController: Controller {
    // GET: Nodes
    public ActionResult Nodes() {
        List < DiagramNode > nodes = new List < DiagramNode > ();
        nodes.Add(new DiagramNode() {
            Id = "node1",
            Width = 100,
            Height = 100,
            OffsetX = 100,
            OffsetY = 100,
            Shape = new { type = "HTML", content= "<div
style='background:#6BA5D7;height:100%;width:100%;'><button type='button'
style='width:100px'> Button</button></div>" },
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

```

Note: HTML node cannot be exported to image format, like JPEG, PNG, and BMP. It is by design, while exporting the diagram is drawn in a canvas. Further, this canvas is exported into image formats. Currently, drawing in a canvas equivalent from all possible HTML is not feasible. Hence, this limitation.

HTML Node With Template

Html elements can be embedded in the diagram using [Html](#) type node. The shape property of the node allows to set the type of node.

HTML_NODE_WITH_TEMPLATE.CSHTML

```

<div class="text-center">

@ (Html.EJS().Diagram("diagram").Width("100%").Height("700px").NodeTemplate("#nodetemplate").Nodes(ViewBag.nodes).Render())

</div>
<script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
</script>

```

Native

Diagram provides support to embed SVG element into a node. The shape property of node allows to set the type of node. To create a [native](#) node, it should be set as **native**.

NATIVE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
    }
}

```

```

public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    nodes.Add(new DiagramNode() {
        Id = "node1",
        Width = 100,
        Height = 100,
        OffsetX = 100,
        OffsetY = 100,
        Shape = new {
            type = "Native",
            content = "<g xmlns=' http:
//www.w3.org/2000/svg'> <g transform='translate(1 1)'><g>          <path
style='fill:#61443C;' d='M61.979,435.057c2.645-0.512,5.291-0.853,7.936-
1.109c-2.01,1.33-4.472,          1.791-6.827,1.28
C62.726,435.13,62.354,435.072,61.979,435.057z' /><path
style='fill:#61443C;' d='M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-
49.749-66.304 c-5.784-1.581-11.753-2.385-17.749-2.389c-2.425-0.028-
4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-23.296 c0.377-47.729-
34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-
2.731,23.296c2.405-0.312,4.829-0.455,7.253-0.427
C472.572,434.123,502.783,464.869,502.469,502.471z' />          </g>          <path
style='fill:#8B685A;' d='M476.869,502.471h7.536c-0.191-32.558,22.574-
60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-
1.28c1.707-0.085,3.413-0.171,5.12-0.171
c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-
1.288,3.414-3.341,3.879-5.685          c7.68-39.706,39.605-70.228,79.616-
76.117c4.325-0.616,8.687-0.929,13.056-0.939c13.281-
0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.
345-0.598,4.337-2.142,5.502-4.264          c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-
0.9,15.666-2.731,23.296          c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389c456.696,444.61,477.033,471.713,4
76.869,502.471          L476.869,502.471z' />          <path style='fill:#66993E;'
d='M502.469,7.537c0,0-6.997,264.96-192.512,252.245c-20.217-1.549-40.166-
5.59-59.392-12.032          c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-
1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z' />          <path
style='fill:#9ACA5C;' d='M476.784,10.183c-1.28,26.197-16.213,238.165-
166.827,249.6          c-20.217-1.549-40.166-5.59-59.392-12.032c-1.365-0.341-
2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144
C238.363,206.279,219.931,41.329,476.784,10.183z' />          <path
style='fill:#66993E;' d='M206.192,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28          c-21.505,7.427-
44.293,10.417-66.987,8.789c21.104,252.103,8.816,94.236,7.621,71.452c-0.085-
1.792-0.085-2.731-0.085-2.731
C222.747,86.129,211.653,216.689,206.192,246.727z' />          <path
style='fill:#9ACA5C;' d='M180.336,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28          c-13.351,4.412-
27.142,7.359-41.131,8.789c21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z' />          </g>          <g>
          <path d='M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-
6.858-1.849-9.298l-8.533-8.533          c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533

```

```
C157.701,425.773,159.872,426.673,162.136,426.671L162.136,426.671z' />
    <path d='M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533    c-3.341-3.281-8.701-3.256-
12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z' />    <path
d='M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-
3.335,8.671-0.054,12.012l8.533,8.533    c3.341,3.281,8.701,3.256,12.012-
0.054c3.311-3.311,3.335-8.671,0.054-12.012L296.169,454.771z' />
    <path d='M386.503,475.37c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533    c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z' />
    <path d='M204.803,409.604c2.264,0.003,4.435-0.897,6.033-
2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012    c-3.311-3.311-8.671-
3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z' />    <path
d='M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012    c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z' />    <path
d='M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012    c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z' />    <path
d='M164.636,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065    c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z' />
    <path d='M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065    c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z' />
    <path d='M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012    c-3.311-3.311-8.671-3.335-
12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z' />    <path
d='M70.77,463.304l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271s3.1,5.28,6.065,6.065    c2.965,0.785,6.122-0.082,8.271-
2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z' />    <path
d='M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065    c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z' />
    <path d='M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-
4.433,0.899-6.033,2.5l-8.533,8.533    c-2.178,2.151-3.037,5.304-
2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-
0.073,8.262-2.251l8.533-8.533    c1.601-1.6,2.5-3.77,2.5-
6.033C204.802,424.408,203.903,422.237,202.302,420.638L202.302,420.638z' />
    <path d='M210.836,463.304c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533
c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-
2.965-0.082-6.122-2.27-8.271L210.836,463.304z' />    <path
d='M343.836,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065    c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z' />
    <path d='M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-
8.671,0.054-12.012l-8.533-8.533    c-3.341-3.281-8.701-3.256-12.012,0.054c-
```

```

3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z' />                <path
d='M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.054l-8.533,8.533c-
2.44,2.441-3.169,6.11-
1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z' />
      <path d='M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z' />
      <path
d='M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62
.581-8.828,88.583-26.796      c94.225-65.725,99.567-227.462,99.75-
234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-1.679-4.141-2.52-6.558-
2.308      C387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881      c-
4.581-7.691-9.996-14.854-16.147-21.358c8.023-38.158,0.241-77.939-21.57-
110.261C160.753,95.829,98.828,68.458,9.228,61.196      c-2.417-0.214-
4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665
c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-5.666-26.723-8.595-40.759-
8.6c-49.449,0.497-91.788,35.567-101.483,84.058      c-5.094-1.093-10.29-1.641-
15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-
0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-
2.499c1.6-1.6,2.499-3.771,2.499-6.035      c0.249-41.103-31.914-75.112-72.967-
77.154c0.65-4.78,0.975-9.598,0.975-14.421c0.914-45.674-28.469-86.455-72.083-
100.045      c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.13
8z M87.519,238.092      c-55.35-38.567-67.358-129.25-69.833-
158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-
162.473-140.577c-2.82-1.165-6.048-0.736-8.466,1.125      s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.77
6,161.392,144.874      C150.795,260.908,115.29,257.451,87.519,238.092z
M279.969,114.046c37.6-53.788,109.708-86.113,214.408-96.138      c-2.65,35.375-
17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-142.305,13.433
c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
6.76c0.404-3.022-0.835-6.031-3.25-7.892      c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z
M262.694,368.017c15.097-26.883,43.468-43.587,74.3-43.746
c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-0.857,14.351-
2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661      c3.899-
29.333,28.777-51.334,58.367-51.617c4-
0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467      c3.858-
2.519,6.531-6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-
70.192c16.045-0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,
368.017L262.694,368.017z' />    </g></g>" },
    }); ViewBag.nodes = nodes;
    return View();
}
}

```

```
}

```

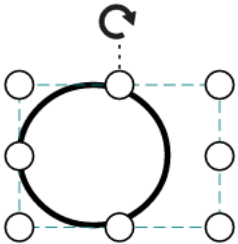
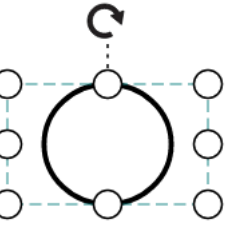
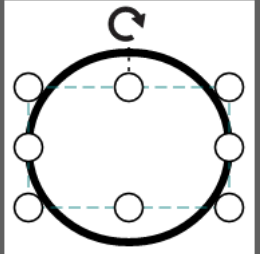
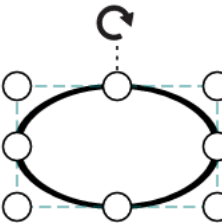
Note: Like HTML node, the native node also cannot be exported to image format. Fill color of native node can be overridden by the inline style or fill of the SVG element specified in the template.

SVG content alignment

Stretch and align the svg content anywhere but within the node boundary.

The scale property of the node allows to stretch the svg content as you desired (either to maintain proportion or to stretch). By default, the `scale` property of node is set as **meet**.

The following tables illustrates all the possible scale options for the node.

Values	Images
None	
Meet	
Slice	
Stretch	

Basic shapes

- The [Basic](#) shapes are common shapes that are used to represent the geometrical information visually. To create basic shapes, the type of the shape should be set as **basic**. Its shape property can be set with any one of the built-in shape.
- To render a rounded rectangle, you need to set the type as basic and shape as rectangle. Set the [cornerRadius](#) property to specify the radius of rounded rectangle.

BASIC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new { type = "Basic", shape = "Rectangle",
cornerRadius = 10 },
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

Note: By default, the shape property of the node is set as **basic**.

Default property for shape is Rectangle.

Note: When the shape is not set for a basic shape, it is considered as a **rectangle**.

The list of basic shapes are as follows.



Rectangle



Ellipse



Triangle



Plus



Star



Pentagon



Heptagon



Octagon



Trapezoid



Decagon



Right
Triangle



Parallelogram

Path

The [Path](#) node is a commonly used basic shape that allows visually to represent the geometrical information. To create a path node, specify the shape as **path**. The path property of node allows you to define the path to be drawn.

PATH.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new { type = "Path", data = "M35.2441,25
L22.7161,49.9937 L22.7161,0.00657536 L35.2441,25 z M22.7167,25 L-
0.00131226,25 M35.2441,49.6337 L35.2441,0.368951 M35.2441,25 L49.9981,25"
            },
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

Flow Shapes

The [flow](#) shapes are used to represent the process flow. It is used for analyzing, designing, and managing for documentation process. To create a flow shape, specify the shape type as **flow**. Flow shapes and by default, it is considered as **process**.

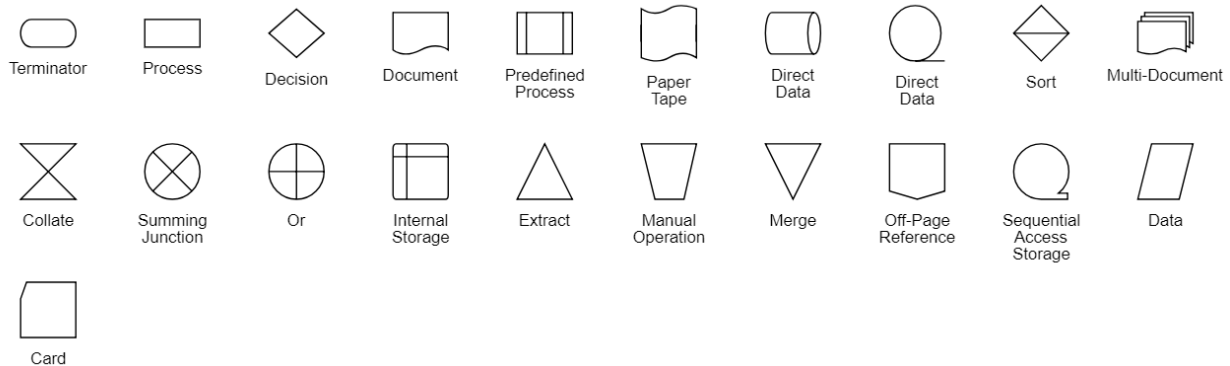
FLOW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
```

```

        Id = "node1",
        Width = 100,
        Height = 100,
        OffsetX = 100,
        OffsetY = 100,
        Shape = new { type = "Flow", shape = "DirectData" },
    });
    ViewBag.nodes = nodes;
    return View();
}
}
}

```



Shapes in Diagram Control

BPMN shapes are used to represent the internal business procedure in a graphical notation and enable you to communicate the procedures in a standard manner. To create a BPMN shape, in the node property shape, type should be set as “bpmn” and its shape should be set as any one of the built-in shapes.

Note: If you want to use BPMN shapes in diagram, you need to inject BpmnDiagrams in the diagram.

END.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers {
    public class NodeController: Controller {
        // GET: Node
        public ActionResult Node() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,

```

```

        OffsetY = 100,
        Shape = new BpmnShapes() {
            Type = "Bpmn",
            Shape = "Event",
            Event = new DiagramBpmnEvent() {
                Event = BpmnEvents.End
            }
        }
    });
    ViewBag.nodes = nodes;
    return View();
}

public class BpmnShapes {
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("event")]
    [JsonProperty("event")]
    public DiagramBpmnEvent Event {
        get;
        set;
    }
}
}
}
}

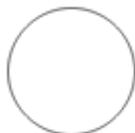
```

Note: The default value for the property shape is "event".

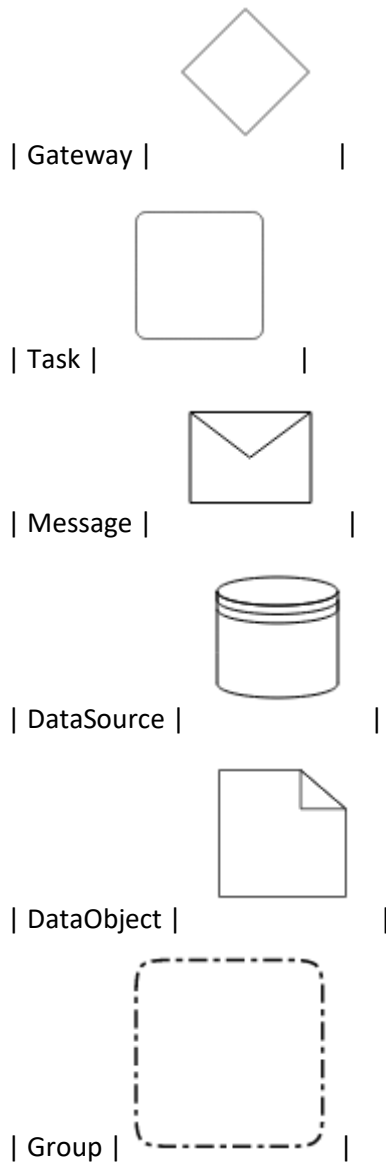
The list of BPMN shapes are as follows:

| Shape | Image |

| ----- | ----- |



| Event | |



The BPMN shapes and its types are explained as follows.

<!-- markdownlint-disable MD033 -->

Event

An [event](#) is notated with a circle and it represents an event in a business process. The type of events are as follows:

- Start
- End
- Intermediate

The event property of the node allows to define the type of the event. The default value of the event is **start**.

TRIGGER.CS












































```

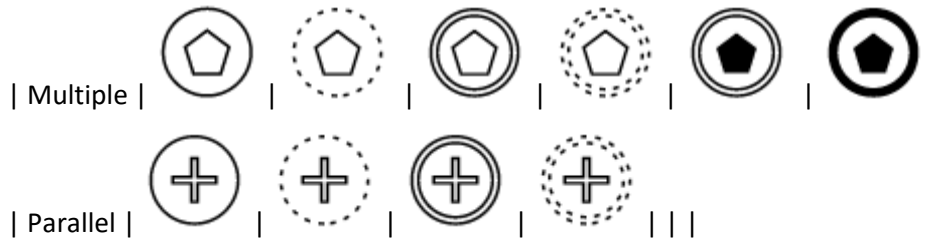
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers {
    public class NodeController: Controller {
        // GET: Node
        public ActionResult Node() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes() {
                    Type = "Bpmn",
                    Shape = "Event",
                    Event = new DiagramBpmnEvent() {
                        Event = BpmnEvents.End, Trigger =
BpmnTriggers.None
                    }
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
        public class BpmnShapes {
            [DefaultValue(null)]
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type {
                get;
                set;
            }
            [DefaultValue(null)]
            [HtmlAttributeName("shape")]
            [JsonProperty("shape")]
            public string Shape {
                get;
                set;
            }
            [DefaultValue(null)]
            [HtmlAttributeName("event")]
            [JsonProperty("event")]
            public DiagramBpmnEvent Event {
                get;
                set;
            }
        }
    }
}

```

Event triggers are notated as icons inside the circle and they represent the specific details of the process. The [trigger](#) property of the node allows you to set the type of trigger and by default, it is set as **none**. The following table illustrates the type of event triggers.

| Triggers | Start | Non-Interrupting Start | Intermediate | Non-Interrupting Intermediate | Throwing Intermediate | End |

	-----	-----	-----	-----	-----	-----	
None							
Message							
Timer							
Conditional							
Link							
Signal							
Error							
Escalation							
Termination							
Compensation							
Cancel							



Gateway

Gateway is used to control the flow of a process and it is represented as a diamond shape. To create a gateway, the shape property of the node should be set as "gateway" and the [gateway](#) property can be set with any of the appropriate gateways.

GATEWAY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers {
    public class NodeController: Controller {
        // GET: Node
        public ActionResult Node() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes() {
                    Type = "Bpmn",
                    Shape = "Gateway",
                    Gateway = new DiagramBpmnGateway() {
                        Type = BpmnGateways.None
                    }
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
        public class BpmnShapes {
            [DefaultValue(null)]
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type {
                get;
                set;
            }
            [DefaultValue(null)]
            [HtmlAttributeName("shape")]






```



```
[JsonProperty("shape")]
public string Shape {
    get;
    set;
}
[DefaultValue(null)]
[HtmlAttributeName("gateway")]
[JsonProperty("gateway")]
public DiagramBpmnGateway Gateway {
    get;
    set;
}
}
```

Note: By default, the gateway will be set as none.

There are several types of gateways as tabulated:

Shape	Image
-----	-----
Exclusive	
Parallel	
Inclusive	
Complex	
EventBased	



| ExclusiveEventBased |



| ParallelEventBased |

Activity

The [activity](#) is the task that is performed in a business process. It is represented by a rounded rectangle.

There are two types of activities. They are listed as follows:

- Task: Occurs within a process and it is not broken down to a finer level of detail.
- Subprocess: Occurs within a process and it is broken down to a finer level of detail.

To create a BPMN activity, set the shape as **activity**. You also need to set the type of the BPMN activity by using the activity property of the node. By default, the type of the activity is set as **task**.

ACTIVITY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.Task
                    }
                }
            });
        }
    }
}
```

```

    }
    });
    ViewBag.nodes = nodes;
    return View();
}
public class BpmnShapes
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}
}

```

The different activities of BPMN process are listed as follows.

Tasks

The [task](#) property of the node allows to define the type of task such as sending, receiving, user based task, etc. By default, the [type](#) property of task is set as **none**. The following code illustrates how to create different types of BPMN tasks. The events property of tasks allow to represent these results as an event attached to the task.

TASK.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers

```

```

{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnTask task = new DiagramBpmnTask();
            task.Type = BpmnTasks.Receive;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.Task,
                        Task = task
                    }
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }

    public class BpmnShapes
    {
        [HtmlAttributeName("type")]
        [JsonProperty("type")]
        public string Type
        {
            get;
            set;
        }

        [HtmlAttributeName("shape")]
        [JsonProperty("shape")]
        public string Shape
        {
            get;
            set;
        }


        [HtmlAttributeName("activity")]
        [JsonProperty("activity")]
        public DiagramBpmnActivity Activity
        {
            get;
            set;
        }
    }
}

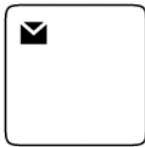
```


```
}  
}
```


The various types of BPMN tasks are tabulated as follows.


Shape	Image
-----	-----

		
Service		

		
Send		

		
Receive		

		
Instantiating Receive		

		
Manual		



| Business Rule |



| User |



| Script |

Subprocess

A [sub-process](#) is a group of tasks, which is used to hide or reveal details of additional levels using the [collapsed](#) property.

SUBPROCESS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnSubProcess SubProcess = new DiagramBpmnSubProcess();
            SubProcess.Collapsed = true;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
```

```

        OffsetX = 100,
        OffsetY = 100,
        Shape = new BpmnShapes()
        {
            Type = "Bpmn",
            Shape = "Activity",
            Activity = new DiagramBpmnActivity()
            {
                Activity = BpmnActivities.SubProcess,
                SubProcess = SubProcess
            }
        }
    });
    ViewBag.nodes = nodes;
    return View();
}
public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }

    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}
}

```

The different types of subprocess are as follows:

- Event subprocess
- Transaction

Event subprocess

A subprocess is defined as an event subprocess, when it is triggered by an event. An event subprocess is placed within another subprocess which is not part of the normal flow of its parent process. You can set

event to a subprocess with the [event](#) and [trigger](#) property of the subprocess. The [type](#) property of subprocess allows you to define the type of subprocess whether it should be event subprocess or transaction subprocess.

SUBEVENT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnSubProcess SubProcess = new DiagramBpmnSubProcess();
            SubProcess.Collapsed = true;
            SubProcess.Type = BpmnSubProcessTypes.Event;
            SubProcess.Events = new List<DiagramBpmnSubEvent>();
            DiagramBpmnSubEvent events = new DiagramBpmnSubEvent();
            events.Event = BpmnEvents.Start;
            events.Trigger = BpmnTriggers.Message;
            SubProcess.Events.Add(events);
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.SubProcess,
                        SubProcess = SubProcess
                    }
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class BpmnShapes
    {
        [HtmlAttributeName("type")]
        [JsonProperty("type")]
    }
}
```



```

        public string Type
        {
            get;
            set;
        }

        [HtmlAttributeName("shape")]
        [JsonProperty("shape")]
        public string Shape
        {
            get;
            set;
        }

        [HtmlAttributeName("activity")]
        [JsonProperty("activity")]
        public DiagramBpmnActivity Activity
        {
            get;
            set;
        }
    }
}

```

Transaction subprocess

- [transaction](#) is a set of activities that logically belong together, in which all contained activities must complete their parts of the transaction; otherwise the process is undone. The execution result of a transaction is one of Successful Completion, Unsuccessful Completion (Cancel), and Hazard (Exception). The **events** property of subprocess allows to represent these results as an event attached to the subprocess.
- The event object allows to define the type of event by which the subprocess will be triggered. The name of the event can be defined to identify the event at runtime.
- The event's offset property is used to set the fraction/ratio (relative to parent) that defines the position of the event shape.
- The trigger property defines the type of the event trigger.
- You can also use define ports and labels to subprocess events by using event's ports and labels properties.

TRANSACTION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{

```

```

public class NodeController : Controller
{
    // GET: Node
    public ActionResult Node()
    {
        List<DiagramNode> nodes = new List<DiagramNode>();
        DiagramBpmnSubProcess SubProcess = new DiagramBpmnSubProcess();
        SubProcess.Collapsed = true;
        SubProcess.Type = BpmnSubProcessTypes.Transaction;
        SubProcess.Events = new List<DiagramBpmnSubEvent>();
        DiagramBpmnSubEvent events = new DiagramBpmnSubEvent();
        events.Event = BpmnEvents.Intermediate;
        events.Trigger = BpmnTriggers.Cancel;
        events.Offset = new DiagramPoint() { X = 0, Y = 0 };
        SubProcess.Events.Add(events);
        DiagramBpmnSubEvent events1 = new DiagramBpmnSubEvent();
        events1.Event = BpmnEvents.Intermediate;
        events1.Trigger = BpmnTriggers.Cancel;
        events1.Offset = new DiagramPoint() { X = 1, Y = 1 };
        SubProcess.Events.Add(events);
        nodes.Add(new DiagramNode()
        {
            Id = "node1",
            Width = 100,
            Height = 100,
            OffsetX = 100,
            OffsetY = 100,
            Shape = new BpmnShapes()
            {
                Type = "Bpmn",
                Shape = "Activity",
                Activity = new DiagramBpmnActivity()
                {
                    Activity = BpmnActivities.SubProcess,
                    SubProcess = SubProcess
                }
            }
        }); ViewBag.nodes = nodes;
        return View();
    }
}

public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }
}

```

```

    }

    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}
}

```

Process

Processes is an array collection that defines the children values for BPMN subprocess.

Loop

[Loop](#) is a task that is internally being looped. The loop property of task allows to define the type of loop. The default value for **loop** is **none**. You can define the loop property in subprocess BPMN shape as shown in the following code.

LOOP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnTask task = new DiagramBpmnTask();
            task.Loop = BpmnLoops.Standard;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.Task,

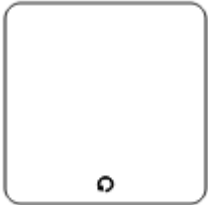
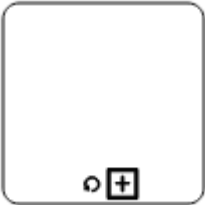
```

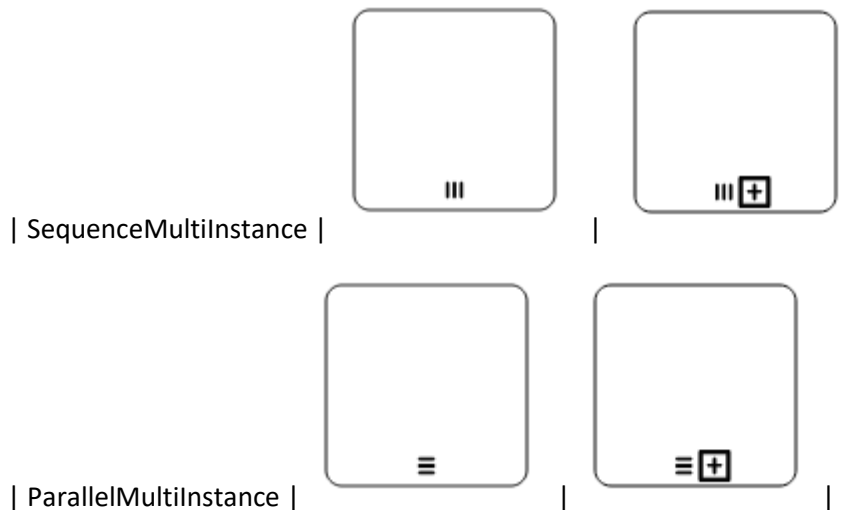
```
Task = task
    }
    }
}); ViewBag.nodes = nodes;
return View();
}
public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }

    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}
```

The following table contains various types of BPMN loops.

Loops	Task	Subprocess
-----	-----	-----
Standard		



Compensation

[Compensation](#) is triggered, when operation is partially failed and enabled it with the compensation property of the task and the subprocess.

COMPENSATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnTask task = new DiagramBpmnTask();
            task.Compensation = true;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.Task,
```

```

        Task = task
    }
    }); ViewBag.nodes = nodes;
    return View();
}
public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }

    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}

```

Call

A [call](#) activity is a global subprocess that is reused at various points of the business flow and set it with the call property of the task.

CALL.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
    }
}

```

```

public ActionResult Node()
{
    List<DiagramNode> nodes = new List<DiagramNode>();
    DiagramBpmnTask task = new DiagramBpmnTask();
    task.Call = true;
    nodes.Add(new DiagramNode()
    {
        Id = "node1",
        Width = 100,
        Height = 100,
        OffsetX = 100,
        OffsetY = 100,
        Shape = new BpmnShapes()
        {
            Type = "Bpmn",
            Shape = "Activity",
            Activity = new DiagramBpmnActivity()
            {
                Activity = BpmnActivities.Task,
                Task = task
            }
        }
    }); ViewBag.nodes = nodes;
    return View();
}

public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }

    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}

```

Adhoc

An adhoc subprocess is a group of tasks that are executed in any order or skipped in order to fulfill the end condition and set it with the [adhoc](#) property of subprocess.

ADHOC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnSubProcess SubProcess = new DiagramBpmnSubProcess();
            SubProcess.Adhoc = true;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.SubProcess,
                        SubProcess = SubProcess
                    }
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
        public class BpmnShapes
        {
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type
            {
                get;
                set;
            }
            [HtmlAttributeName("shape")]
            [JsonProperty("shape")]
        }
    }
}
```



```

        public string Shape
        {
            get;
            set;
        }
        [HtmlAttributeName("activity")]
        [JsonProperty("activity")]
        public DiagramBpmnActivity Activity
        {
            get;
            set;
        }
    }
}

```

Boundary

Boundary represents the type of task that is being processed. The [boundary](#) property of subprocess allows to define the type of boundary. By default, it is set as **default**.

BOUNDARY.CS

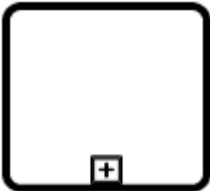

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnSubProcess SubProcess = new DiagramBpmnSubProcess();
            SubProcess.Boundary = BpmnBoundary.Call;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.SubProcess,
                        SubProcess = SubProcess
                    }
                }
            });
        }
    }
}

```

```
        }
    }
    });
    ViewBag.nodes = nodes;
    return View();
}
public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }
    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
}
```

The following table contains various types of BPMN boundaries.

Boundary	Image
-----	-----
Call	
Event	



| Default |

Data

A data object represents information flowing through the process, such as data placed into the process, data resulting from the process, data that needs to be collected, or data that must be stored. To define a [data object](#), set the shape as **DataObject** and the type property defines whether data is an input or an output. You can create multiple instances of data object with the collection property of data.

DATAOBJECT.CS

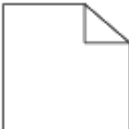

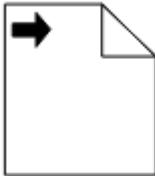
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            DiagramBpmnTask task = new DiagramBpmnTask();
            task.Compensation = true;
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Activity",
                    Activity = new DiagramBpmnActivity()
                    {
                        Activity = BpmnActivities.Task,
                        Task = task
                    }
                }
            }); ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

```
public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }

    [HtmlAttributeName("activity")]
    [JsonProperty("activity")]
    public DiagramBpmnActivity Activity
    {
        get;
        set;
    }
}
```

The following table contains various representation of BPMN data object.

Boundary Image		
----- -----		
Collection Data Object		
Data Input		
Data Output		

Datasource

Datasource is used to store or access data associated with a business process. To create a datasource, set the shape as **datasource**.

DATASOURCE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "DataSource",
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
        public class BpmnShapes
        {
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type
            {
                get;
                set;
            }
            [HtmlAttributeName("shape")]
            [JsonProperty("shape")]
            public string Shape
            {
                get;
                set;
            }
            [HtmlAttributeName("dataObject")]

```

```

        [JsonProperty("dataObject")]
        public DiagramBpmnDataObject DataObject
        {
            get;
            set;
        }
    }
}

```

Artifact

Artifact is used to show additional information about a process in order to make it easier to understand. There are two types of artifacts in BPMN.

- Text annotation
- Group

Text annotation

- A BPMN object can be associated with a text annotation which does not affect the flow but gives details about objects within a flow. The annotation property of the node is used to connect an annotation element to the BPMN node.
- The annotation element can be displaced into a different position interactively by dragging the annotation to a particular position.
- The annotation element can be switched from a BPMN node to another BPMN node simply by dragging the source end of the annotation connector into the other BPMN node.
- The annotation angle property is used to set the angle between the BPMN shape and the annotation.
- The annotation direction property is used to set the direction of the text annotation.
- To set the size for text annotation, use width and height properties.
- The annotation length property is used to set the distance between the BPMN shape and the annotation.
- The annotation text property defines the additional information about the flow object in a BPMN process.

TEXTANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
    }
}

```

```

public ActionResult Node()
{
    List<DiagramNode> nodes = new List<DiagramNode>();
    DiagramBpmnDataObject dataObject = new DiagramBpmnDataObject();
    dataObject.Collection = true;
    dataObject.Type = BpmnDataObjects.Input;
    List<DiagramBpmnAnnotation> annotation = new
List<DiagramBpmnAnnotation>();
    DiagramBpmnAnnotation label = new DiagramBpmnAnnotation();
    label.Angle = 45;
    label.Length = 150;
    label.Text = "Left";
    label.Id = "left";
    annotation.Add(label);
    nodes.Add(new DiagramNode()
    {
        Id = "node1",
        Width = 100,
        Height = 100,
        OffsetX = 100,
        OffsetY = 100,
        Shape = new BpmnShapes()
        {
            Type = "Bpmn",
            Shape = "DataObject",
            DataObject = dataObject,
            Annotations = annotation
        }
    });
    ViewBag.nodes = nodes;
    return View();
}

public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }

    [HtmlAttributeName("annotations")]
    [JsonProperty("annotations")]
    public List<DiagramBpmnAnnotation> Annotations
    {
        get;
        set;
    }

    [HtmlAttributeName("dataObject")]
    [JsonProperty("dataObject")]

```

```

        public DiagramBpmnDataObject DataObject
        {
            get;
            set;
        }
    }
}

```

Group

A group is used to frame a part of the diagram, shows that elements included in it logically belong together and does not have any other semantics other than organizing elements. To create a group, the shape property of the node should be set as **group**.

GROUP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            nodes.Add(new DiagramNode()
            {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
                Shape = new BpmnShapes()
                {
                    Type = "Bpmn",
                    Shape = "Group",
                }
            });
            ViewBag.nodes = nodes;
            return View();
        }
        public class BpmnShapes
        {
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type
            {
                get;
            }
        }
    }
}

```



```

        set;
    }
    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }
}
}
}

```

BPMN flows

BPMN Flows are lines that connects BPMN flow objects.

Association

BPMN Association flow is used to link flow objects with its corresponding text or artifact. An association is represented as a dotted graphical line with opened arrow. The types of association are as follows:

- Directional
- BiDirectional
- Default

The association property allows you to define the type of association.

ASSOCIATION.CS




```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector()
            {
                Id = "connector1",
                SourcePoint = new DiagramPoint() { X=100,Y=100},
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Type= Segments.Orthogonal,
                Shape=new BpmnShapes()
            }
        }
    }
}

```

```
        {
            Type="Bpmn",
            Flow= "Association",
            Association= "BiDirectional"
        }
    });
    ViewBag.connectors = Connectors;
    return View();
}
public class BpmnShapes
{
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [HtmlAttributeName("shape")]
    [JsonProperty("shape")]
    public string Shape
    {
        get;
        set;
    }
    [HtmlAttributeName("flow")]
    [JsonProperty("flow")]
    public string Flow
    {
        get;
        set;
    }
    [HtmlAttributeName("association")]
    [JsonProperty("association")]
    public string Association
    {
        get;
        set;
    }
}
}
```

The following table demonstrates the visual representation of association flows.

Association	Image
Default	
Directional	
BiDirectional	

Note: The default value for the property association is **default**.

Sequence

A **Sequence** flow shows the order in which the activities are performed in a BPMN process and is represented by a solid graphical line. The types of sequence are as follows:

- Normal
- Conditional
- Default

The sequence property allows to define the type of sequence.

SEQUENCE.CS


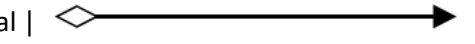
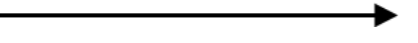
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector()
            {
                Id = "connector1",
                SourcePoint = new DiagramPoint() { X=100,Y=100},
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Type= Segments.Orthogonal,
                Shape=new BpmnShapes()
                {
                    Type="Bpmn",
                    Flow= "Sequence",
                    Sequence = "Conditional"
                }
            });
            ViewBag.connectors = Connectors;
            return View();
        }
        public class BpmnShapes
        {
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type
            {
                get;
                set;
            }
        }
    }
}
```

```

        [HtmlAttributeName("shape")]
        [JsonProperty("shape")]
        public string Shape
        {
            get;
            set;
        }
        [HtmlAttributeName("flow")]
        [JsonProperty("flow")]
        public string Flow
        {
            get;
            set;
        }
        [HtmlAttributeName("sequence")]
        [JsonProperty("sequence")]
        public string Sequence
        {
            get;
            set;
        }
    }
}

```

The following table contains various representation of sequence flows.

Sequence	Image
-----	-----
Default	
Conditional	
Normal	

Note: The default value for the property sequence is **normal**.

Message

A **Message** flow shows the flow of messages between two participants and is represented by dashed line. The types of message are as follows:

- InitiatingMessage
- NonInitiatingMessage
- Default

The message property allows to define the type of message.

MESSAGE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

```




```

using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector()
            {
                Id = "connector1",
                SourcePoint = new DiagramPoint() { X=100,Y=100},
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Type= Segments.Orthogonal,
                Shape=new BpmnShapes()
                {
                    Type="Bpmn",
                    Flow= "Message",
                    Message = "InitiatingMessage"
                }
            });
            ViewBag.connectors = Connectors;
            return View();
        }
        public class BpmnShapes
        {
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type
            {
                get;
                set;
            }
            [HtmlAttributeName("shape")]
            [JsonProperty("shape")]
            public string Shape
            {
                get;
                set;
            }
            [HtmlAttributeName("message")]
            [JsonProperty("message")]
            public string Message
            {
                get;
                set;
            }
            [HtmlAttributeName("flow")]
            [JsonProperty("flow")]
            public string Flow
            {
                get;
            }
        }
    }
}

```



The following table contains various representation of message flows.

Message	Image
-----	-----
Default	
InitiatingMessage	
NonInitiatingMessage	

Note: The default value for the property `message` is `default`.

UML Diagram Shapes

UML Class Diagram

A class diagram visually depicts the static structure of an application and is extensively employed in modeling object-oriented systems. It holds a unique position in UML diagrams, as it directly aligns with object-oriented languages. The diagram also facilitates automatic generation of class diagram shapes based on business logic, streamlining the translation from conceptual models to practical implementation.

Uml Class Diagram Shapes

The UML class diagram shapes are explained as follows.

Class

- A class defines a group of objects that share common specifications, features, constraints, and semantics. To create a class object, the classifier should be defined using the [class](#) notation. This notation serves as a foundational element in object-oriented programming, encapsulating the essential characteristics and behavior that objects belonging to the class will exhibit.
- Also, define the [name](#), [attributes](#), and [methods](#) of the class using the class property of node.
- The attribute's [name](#), [type](#), and [scope](#) properties allows to define the name, data type, and visibility of the attribute.
- The method's [name](#), [parameters](#), [type](#), and [scope](#) properties allows to define the name, parameter, return type, and visibility of the methods.
- The method parameters object properties allows to define the name and type of the parameter.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
  <div class="content-wrapper">
    @(Html.EJS().Diagram("container")
      .Width("100%")
      .Height("500px")
      .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
      .Nodes(ViewBag.nodes).Render())
  
```

</div>

</div>

CLASS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<UMLProperty> patientProperty = new List<UMLProperty>();
            patientProperty.Add(CreateUMLProperty("accepted", "Date"));
            List<UMLMethods> patientMethods = new List<UMLMethods>();
            patientMethods.Add(CreateUMLMethod("getHistory", "History"));
            nodes.Add(
                new DiagramNode()
                {
                    Id = "Patient",
                    OffsetX = 200,
                    OffsetY = 250,
                    Shape = new UmlClassifierShapeModel()
                    {
                        Type = "UMLClassifier",
                        Class = new Class()
                        {
                            Name = "Patient",
                            Attributes = patientProperty,
                            Methods = patientMethods
                        },
                    },
                },
            );
            ViewBag.Nodes = nodes;
            return View();
        }
        public UMLProperty CreateUMLProperty(string name, string type)
        {
            UMLProperty property = new UMLProperty();
            property.Name = name;
            property.Type = type;
            return property;
        }
        public UMLMethods CreateUMLMethod(string name, string type)
        {
            UMLMethods method = new UMLMethods();
            method.Name = name;
        }
    }
}

```

```
        method.Type = type;
        return method;
    }
}

public class UmlClassifierShapeModel
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("class")]
    [JsonProperty("class")]
    public Class Class
    {
        get;
        set;
    }
}

public class Class
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("attributes")]
    [JsonProperty("attributes")]
    public List<UMLProperty> Attributes
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("methods")]
    [JsonProperty("methods")]
    public List<UMLMethods> Methods
    {
        get;
        set;
    }
}

public class UMLProperty
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {

```




Interface

- An interface is a specific type of classifier that signifies a declaration of a cohesive set of public features and obligations. When creating an interface, it involves defining the classifier property using the [interface](#) notation. This essential concept in object-oriented programming outlines a contract for classes to adhere to, specifying the required methods and behaviors without delving into the implementation details.
- Also, define the [name](#), [attributes](#), and [methods](#) of the interface using the interface property of the node.
- The attribute's name, type, and scope properties allows to define the name, data type, and visibility of the attribute.
- The method's name, parameter, type, and scope properties allows to define the name, parameter, return type, and visibility of the methods.
- The method parameter object properties of name and type allows to define the name and type of the parameter.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">

```

```

        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

INTERFACE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<UMLProperty> patientProperty = new List<UMLProperty>();
            patientProperty.Add(CreateUMLProperty("accepted", "Date"));
            List<UMLMethods> patientMethods = new List<UMLMethods>();
            patientMethods.Add(CreateUMLMethod("getHistory", "History"));
            nodes.Add(
                new DiagramNode()
                {
                    Id = "Patient",
                    OffsetX = 200,
                    OffsetY = 250,
                    Shape = new UmlClassifierShapeModel()
                    {
                        Type = "UMLClassifier",
                        Interface = new Interface()
                        {
                            Name = "Patient",
                            Attributes = patientProperty,
                            Methods = patientMethods
                        },
                    },
                },
            );
            ViewBag.Nodes = nodes;
            return View();
        }
        public UMLProperty CreateUMLProperty(string name, string type)
        {
            UMLProperty property = new UMLProperty();
            property.Name = name;
            property.Type = type;
            return property;
        }
    }
}

```

```

    }
    public UMLMethods CreateUMLMethod(string name, string type)
    {
        UMLMethods method = new UMLMethods();
        method.Name = name;
        method.Type = type;
        return method;
    }
}

public class UmlClassifierShapeModel
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [DefaultValue(null)]
    [HtmlAttributeName("class")]
    [JsonProperty("class")]
    public Class Class
    {
        get;
        set;
    }
}

public class Interface
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }

    [DefaultValue(null)]
    [HtmlAttributeName("attributes")]
    [JsonProperty("attributes")]
    public List<UMLProperty> Attributes
    {
        get;
        set;
    }

    [DefaultValue(null)]
    [HtmlAttributeName("methods")]
    [JsonProperty("methods")]
    public List<UMLMethods> Methods
    {
        get;
        set;
    }
}

public class UMLProperty
{

```

```

        [DefaultValue (null)]
        [HtmlAttributeName ("name")]
        [JsonProperty ("name")]
        public string Name
        {
            get;
            set;
        }
        [DefaultValue (null)]
        [HtmlAttributeName ("type")]
        [JsonProperty ("type")]
        public string Type
        {
            get;
            set;
        }
    }
    public class UMLMethods
    {
        [DefaultValue (null)]
        [HtmlAttributeName ("name")]
        [JsonProperty ("name")]
        public string Name
        {
            get;
            set;
        }
        [DefaultValue (null)]
        [HtmlAttributeName ("type")]
        [JsonProperty ("type")]
        public string Type
        {
            get;
            set;
        }
    }
}

```

Enumeration

- To establish an enumeration, designate the classifier property of the node as [enumeration](#). Additionally, define the name and enumerate the members of the enumeration using the appropriate enumeration property of the node. This process encapsulates a set of distinct values within the enumeration, allowing for a clear representation of specific, named constants within a system.
- You can set a name for the enumeration members collection using the name property of members collection.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @ (Html.EJS () .Diagram ("container")
        .Width ("100%")

```

```

.Height("500px")
.Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
.Nodes(ViewBag.nodes).Render()
</div>
</div>

```

ENUMERATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<UMLMembers> patientMembers = new List<UMLMembers>();
            patientMembers.Add(CreateUMLMembers("Checking Account"));
            patientMembers.Add(CreateUMLMembers("Savings Account"));
            patientMembers.Add(CreateUMLMembers("Credit Account"));
            nodes.Add(
                new DiagramNode()
                {
                    Id = "Patient",
                    OffsetX = 200,
                    OffsetY = 250,
                    Shape = new UmlClassifierShapeModel()
                    {
                        Type = "UMLClassifier",
                        Enumeration = new Enumeration()
                        {
                            Name = "AccountType",
                            Members = patientMembers
                        },
                    },
                },
            );
            ViewBag.Nodes = nodes;
            return View();
        }
        public UMLMembers CreateUMLMembers(string name)
        {
            UMLMembers members = new UMLMembers();
            members.Name = name;
            return members;
        }
    }
    public class UmlClassifierShapeModel
    {

```

```

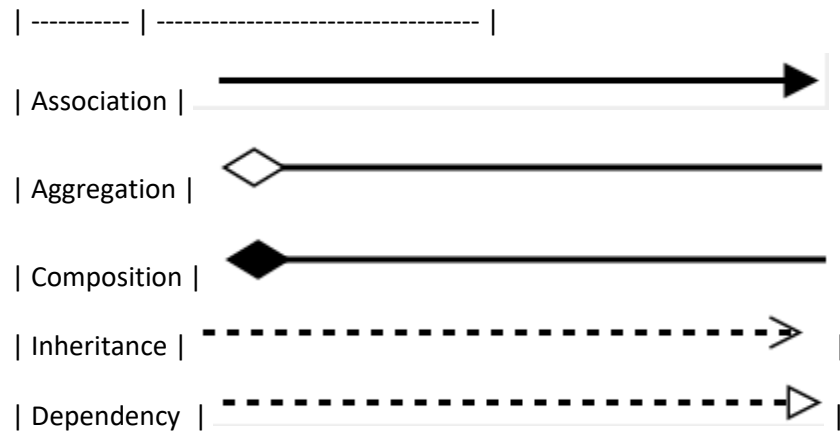
        [DefaultValue (null)]
        [HtmlAttributeName ("type")]
        [JsonProperty ("type")]
        public string Type
        {
            get;
            set;
        }
        [DefaultValue (null)]
        [HtmlAttributeName ("class")]
        [JsonProperty ("class")]
        public Class Class
        {
            get;
            set;
        }
    }
    public class Enumeration
    {
        [DefaultValue (null)]
        [HtmlAttributeName ("name")]
        [JsonProperty ("name")]
        public string Name
        {
            get;
            set;
        }
        [DefaultValue (null)]
        [HtmlAttributeName ("members")]
        [JsonProperty ("members")]
        public List<UMLMembers> Members
        {
            get;
            set;
        }
    }
    public class UMLMembers
    {
        [DefaultValue (null)]
        [HtmlAttributeName ("name")]
        [JsonProperty ("name")]
        public string Name
        {
            get;
            set;
        }
    }
}

```

UML Class Relationships

- A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

| Shape | Image |



Association

Association is basically a set of links that connects elements of an UML model. The type of association are as follows.

1. Directional
2. BiDirectional

The association property allows to define the type of association. The default value of association is "Directional".

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
  <div class="content-wrapper">
    @(Html.EJS().Diagram("container")
      .Width("100%")
      .Height("500px")
      .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
      .Connectors(ViewBag.connectors).Render())
  </div>
</div>
```

ASSOCIATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
        }
    }
}
```

```

        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() {
            Id = "connector",
            SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
            TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
            Shape = new {type = "UmlClassifier",
relationship="Association", association="BiDirectional"}
        });
        ViewBag.connectors = Connectors;
        return View();
    }
}

```

Aggregation

Aggregation is a binary association between a property and one or more composite objects which group together a set of instances. Aggregation is decorated with a hollow diamond. To create an aggregation shape, define the relationship as “aggregation”.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

AGGREGATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() {
                Id = "connector",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },

```



```

        Shape = new {type = "UmlClassifier",
relationship="Aggregation" }
    });
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

Composition

Composition is a “strong” form of “aggregation”. Composition is decorated with a black diamond. To create a composition shape, define the relationship property of connector as “composition”.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

COMPOSITION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() {
                Id = "connector",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Shape = new {type = "UmlClassifier",
relationship="Composition"}
            });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

```
}
```

Dependency

Dependency is a directed relationship, which is used to show that some UML elements needs or depends on other model elements for specifications. Dependency is shown as dashed line with opened arrow. To create a dependency, define the relationship property of connector as “dependency”.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>
```

DEPENDENCY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramConnector> Connectors = new
            List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() {
                Id = "connector",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Shape = new {type = "UmlClassifier",
                relationship="Dependency"}
            });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}
```

Inheritance

Inheritance is also called as “generalization”. Inheritance is a binary taxonomic directed relationship between a more general classifier (super class) and a more specific classifier (subclass). Inheritance is shown as a line with hollow triangle.

To create an inheritance, define the relationship as “inheritance”.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>
```

INHERITANCE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramConnector> Connectors = new
            List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() {
                Id = "connector",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Shape = new {type = "UmlClassifier",
                relationship="Inheritance"}
            });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}
```

Multiplicity

Multiplicity is a definition of an inclusive interval of non-negative integers to specify the allowable number of instances of described element. The type of multiplicity are as follows.

1. OneToOne
2. ManyToOne
3. OneToMany
4. ManyToMany
 - By default the multiplicity will be considered as “OneToOne”.
 - The multiplicity property in UML allows to specify large number of elements or some collection of elements.
 - The shape multiplicity’s source property is used to set the source label to connector and the target property is used to set the target label to connector.
 - To set an optionality or cardinality for the connector source label, use optional property.
 - The [lowerBounds](#) and [upperBounds](#) could be natural constants or constant expressions evaluated to natural (non negative) number. Upper bound could be also specified as asterisk ‘*’ which denotes unlimited number of elements. Upper bound should be greater than or equal to the lower bound.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @ (Html.EJS () .Diagram ("container")
        .Width ("100%")
        .Height ("500px")
        .Mode (Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors (ViewBag.connectors) .Render () )
    </div>
</div>
```

MULTIPLICITY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramConnector> Connectors = new
            List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() {
                Id = "connector",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Shape = new {type = "UmlClassifier",
                relationship="Dependency", multiplicity= new {type="OneToMany"}}
            });
        }
    }
}
```

```

        ViewBag.connectors = Connectors;
        return View();
    }
}

```

How to add UML child at runtime

In UML nodes, child elements such as member, method and attribute can be added either programmatically or interactively.

Adding UML child through code

The [addChildToUmlNode](#) method is employed for dynamically adding a child to the UML node during runtime, providing flexibility in modifying the diagram structure programmatically.

The following code illustrates how to add methods to UML nodes in diagram.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <button type="button" id="addmethod">addmethod</button>
    <div class="content-wrapper">
        @ (Html.EJS().Diagram("container")
            .Width("100%")
            .Height("500px")
            .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
            .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

METHOD.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<UMLProperty> patientProperty = new List<UMLProperty>();
            patientProperty.Add(CreateUMLProperty("accepted", "Date"));
            List<UMLMethods> patientMethods = new List<UMLMethods>();
            patientMethods.Add(CreateUMLMethod("getHistory", "History"));
            nodes.Add(
                new DiagramNode()
                {
                    Id = "Patient",
                    OffsetX = 200,
                    OffsetY = 250,
                    Shape = new UmlClassifierShapeModel()
                }
            );
        }
    }
}

```

```

        {
            Type = "UMLClassifier",
            Interface = new Interface()
            {
                Name = "Patient",
                Attributes = patientProperty,
                Methods = patientMethods
            },
        },
    },
);
ViewBag.Nodes = Nodes
return View();
}
public UMLProperty CreateUMLProperty(string name, string type)
{
    UMLProperty property = new UMLProperty();
    property.Name = name;
    property.Type = type;
    return property;
}
public UMLMethods CreateUMLMethod(string name, string type)
{
    UMLMethods method = new UMLMethods();
    method.Name = name;
    method.Type = type;
    return method;
}
}
public class UmlClassifierShapeModel
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("class")]
    [JsonProperty("class")]
    public Class Class
    {
        get;
        set;
    }
}
public class Interface
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
}

```

```

    }
    [DefaultValue(null)]
    [HtmlAttributeName("attributes")]
    [JsonProperty("attributes")]
    public List<UMLProperty> Attributes
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("methods")]
    [JsonProperty("methods")]
    public List<UMLMethods> Methods
    {
        get;
        set;
    }
}
public class UMLProperty
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
}
public class UMLMethods
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
}
}

```

```
`ts
```

```
let node = diagram.nodes[0];
```

```
let methods = { name: 'getHistory', style: { color: "red", }, parameters: [{ name: 'Date', style: {} }], type: 'History' };
```

```
diagram.addChildToUmlNode(node, methods, 'Methods');
```

```
,
```

The following code illustrates how to add attributes to UML nodes in diagram.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <button type="button" id="addattribute">addattribute</button>
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

ATTRIBUTE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<UMLProperty> patientProperty = new List<UMLProperty>();
            patientProperty.Add(CreateUMLProperty("accepted", "Date"));
            List<UMLMethods> patientMethods = new List<UMLMethods>();
            patientMethods.Add(CreateUMLMethod("getHistory", "History"));
            nodes.Add(
                new DiagramNode()
                {
                    Id = "Patient",
                    OffsetX = 200,
                    OffsetY = 250,
                    Shape = new UmlClassifierShapeModel()
                    {
                        Type = "UMLClassifier",
                        Interface = new Interface()
                    }
                }
            );
        }
    }
}
```



```

        {
            Name = "Patient",
            Attributes = patientProperty,
            Methods = patientMethods
        },
    },
}

);
ViewBag.Nodes = Nodes
return View();
}

public UMLProperty CreateUMLProperty(string name, string type)
{
    UMLProperty property = new UMLProperty();
    property.Name = name;
    property.Type = type;
    return property;
}

public UMLMethods CreateUMLMethod(string name, string type)
{
    UMLMethods method = new UMLMethods();
    method.Name = name;
    method.Type = type;
    return method;
}
}

public class UmlClassifierShapeModel
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }

    [DefaultValue(null)]
    [HtmlAttributeName("class")]
    [JsonProperty("class")]
    public Class Class
    {
        get;
        set;
    }
}

public class Interface
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }

    [DefaultValue(null)]
    [HtmlAttributeName("attributes")]

```

```

        [JsonProperty("attributes")]
        public List<UMLProperty> Attributes
        {
            get;
            set;
        }
        [DefaultValue(null)]
        [HtmlAttributeName("methods")]
        [JsonProperty("methods")]
        public List<UMLMethods> Methods
        {
            get;
            set;
        }
    }
    public class UMLProperty
    {
        [DefaultValue(null)]
        [HtmlAttributeName("name")]
        [JsonProperty("name")]
        public string Name
        {
            get;
            set;
        }
        [DefaultValue(null)]
        [HtmlAttributeName("type")]
        [JsonProperty("type")]
        public string Type
        {
            get;
            set;
        }
    }
    public class UMLMethods
    {
        [DefaultValue(null)]
        [HtmlAttributeName("name")]
        [JsonProperty("name")]
        public string Name
        {
            get;
            set;
        }
        [DefaultValue(null)]
        [HtmlAttributeName("type")]
        [JsonProperty("type")]
        public string Type
        {
            get;
            set;
        }
    }
}

```

`ts

```
let node = diagram.selectedItems.nodes[0];
let attributes = { name: 'accepted', type: 'Date', style: { color: "red", } };
diagram.addChildToUmlNode(node, attributes, "Attributes");
,
```

The following code illustrates how to add members to UML nodes in diagram.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <button type="button" id="addmember">addmember</button>
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

MEMBER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            List<DiagramNode> nodes = new List<DiagramNode>();
            List<UMLMembers> patientMembers = new List<UMLMembers>();
            patientMembers.Add(CreateUMLMembers("Checking Account"));
            patientMembers.Add(CreateUMLMembers("Savings Account"));
            patientMembers.Add(CreateUMLMembers("Credit Account"));
            nodes.Add(
                new DiagramNode()
                {
                    Id = "Patient",
                    OffsetX = 200,
                    OffsetY = 250,
                    Shape = new UmlClassifierShapeModel()
                    {
                        Type = "UMLClassifier",
                        Enumeration = new Enumeration()
                        {
                            Name = "AccountType",
                            Members = patientMembers
                        }
                    },
                },
            );
        }
    }
}
```

```

        },

        );
        ViewBag.Nodes = Nodes
        return View();
    }
    public UMLMembers CreateUMLMembers(string name)
    {
        UMLMembers members = new UMLMembers();
        members.Name = name;
        return members;
    }
}

public class UmlClassifierShapeModel
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("class")]
    [JsonProperty("class")]
    public Class Class
    {
        get;
        set;
    }
}

public class Enumeration
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("members")]
    [JsonProperty("members")]
    public List<UMLMembers> Members
    {
        get;
        set;
    }
}

public class UMLMembers
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name

```



`ts

```
let node = diagram.selectedItems.nodes[0];
```

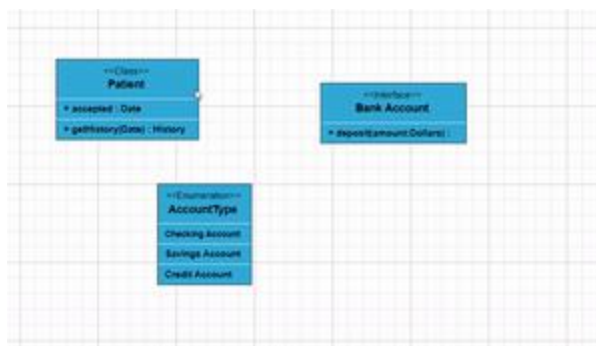
```
let members = { name: 'Checking new', style: { color: "red", }, isSeparator: true };
```

```
diagram.addChildToUmlNode(node, members, "Members");
```

,

Adding UML child through user interaction

To include a child, select a node, move the mouse outside it, and position the pointer near the right side. A highlighter emerges between the two child elements. Click the highlighter to add a child type to the chosen UML node seamlessly. The following gif illustrates how to add Child through user interaction.



Adding UML Nodes in Symbol palette

UML built-in shapes are efficiently rendered in a symbol palette. The `symbols` property is utilized to define UML symbols with the necessary classes and methods. By incorporating this feature, you can seamlessly augment the palette with a curated collection of predefined UML symbols, thereby enhancing the versatility of your UML diagramming application.

The following code example showcases the rendering of UML built-in shapes in a symbol palette.

CSHTML

```

<div id="umlActivityDiagram" style="width: 100%; height: 521px">
    <div id="palette-space" class="sb-mobile-palette">

@ (Html.EJS () .SymbolPalette ("symbolPalette") .Palettes (ViewBag.Palette) .GetSymbolInfo ("getSymbolInfo") .ExpandMode (ExpandMode.Multiple) .Width ("100%") .Height ("700px")
        .SymbolMargin (new SymbolPaletteMargin () { Left = 15, Right = 15, Top = 15, Bottom = 15 }) .SymbolHeight (60) .SymbolWidth (60)
        .GetNodeDefaults ("getSymbolNodes") .Render ()
    </div>
    <div id="diagram-space" class="sb-mobile-diagram">

@ (Html.EJS () .Diagram ("diagram") .Width ("100%") .Height ("700px") .Render ()

```

```

    )
  </div>
</div>

```

UMLCLASS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<Syncfusion.EJ2.Diagrams.DiagramNode> umlShapes = new
List<Syncfusion.EJ2.Diagrams.DiagramNode>();
            umlShapes.Add(new DiagramNode() {
                Id = "Class",
                OffsetX = 200,
                OffsetY = 250,
                Shape = new UmlClassifierShapeModel()
                {
                    Type = "UmlClassifier",
                    ClassShapes = new ClassShapes()
                    {
                        Name = "Patient",
                    },
                    Classifier = "Class"
                },
            });
            umlShapes.Add(new DiagramNode() {
                Id = "Interface",
                OffsetX = 400,
                OffsetY = 350,
                Shape = new UmlClassifierShapeModel()
                {
                    Type = "UmlClassifier",
                    InterfaceShapes = new InterfaceShapes()
                    {
                        Name = "Bank Account",
                    },
                    Classifier = "Interface"
                },
            });
            umlShapes.Add(new DiagramNode() {
                Id = "Enumeration",
                OffsetX = 600,
                OffsetY = 450,
                Shape = new UmlClassifierShapeModel()
                {
                    Type = "UmlClassifier",
                    Enumerations = new Enumerations()
                    {

```

```

        Name = "AccountType",
    },
    Classifier = "Enumeration"
    },
    });
    List<SymbolPalettePalette> Palettes = new
List<SymbolPalettePalette>();
    Palettes.Add(new SymbolPalettePalette() { Id = "UMLClass",
Expanded = true, Symbols = umlShapes, Title = "UML Nodes" });
    ViewBag.palettes = Palettes;
    ViewBag.getSymbolInfo = "getSymbolInfo";
    ViewBag.getSymbolDefaults = "getSymbolDefaults";
    return View();
}
}
public class UmlClassifierShapeModel
{
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("ClassShapes")]
    [JsonProperty("classShape")]
    public ClassShapes ClassShapes
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("Enumerations")]
    [JsonProperty("enumeration")]
    public Enumerations Enumerations
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("InterfaceShapes")]
    [JsonProperty("interfaceShape")]
    public InterfaceShapes InterfaceShapes
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("classifier")]
    [JsonProperty("classifier")]
    public string Classifier
    {
        get;
        set;
    }
}

```

```
}  
}  
public class Enumerations  
{  
    [DefaultValue(null)]  
    [HtmlAttributeName("name")]  
    [JsonProperty("name")]  
    public string Name  
    {  
        get;  
        set;  
    }  
    [DefaultValue(null)]  
    [HtmlAttributeName("members")]  
    [JsonProperty("members")]  
    public List<UMLMembers> Members  
    {  
        get;  
        set;  
    }  
}  
public class ClassShapes  
{  
    [DefaultValue(null)]  
    [HtmlAttributeName("name")]  
    [JsonProperty("name")]  
    public string Name  
    {  
        get;  
        set;  
    }  
    [DefaultValue(null)]  
    [HtmlAttributeName("attributes")]  
    [JsonProperty("attributes")]  
    public List<UMLProperty> Attributes  
    {  
        get;  
        set;  
    }  
    [DefaultValue(null)]  
    [HtmlAttributeName("methods")]  
    [JsonProperty("methods")]  
    public List<UMLMethods> Methods  
    {  
        get;  
        set;  
    }  
}  
public class InterfaceShapes  
{  
    [DefaultValue(null)]  
    [HtmlAttributeName("name")]  
    [JsonProperty("name")]  
    public string Name  
    {  
        get;  
        set;
```



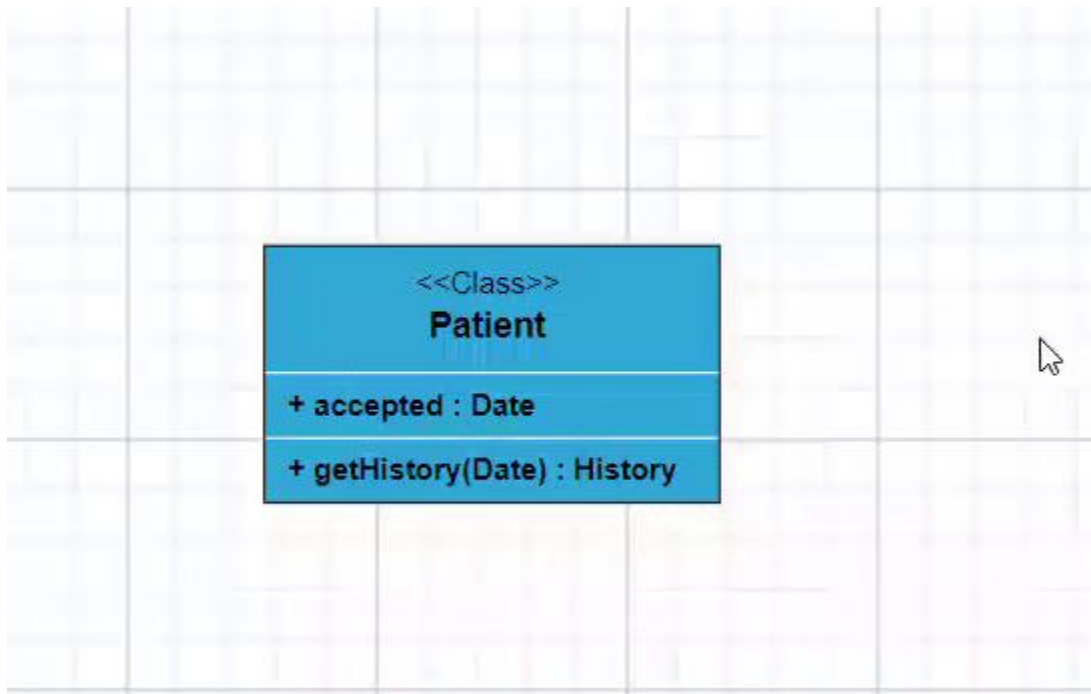
```
}
[DefaultValue(null)]
[HtmlAttributeName("attributes")]
[JsonProperty("attributes")]
public List<UMLProperty> Attributes
{
    get;
    set;
}
[DefaultValue(null)]
[HtmlAttributeName("methods")]
[JsonProperty("methods")]
public List<UMLMethods> Methods
{
    get;
    set;
}
}
public class UMLProperty
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
}

public class UMLMethods
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("type")]
    [JsonProperty("type")]
    public string Type
    {
        get;
        set;
    }
}
```

```
public class UMLMembers
{
    [DefaultValue(null)]
    [HtmlAttributeName("name")]
    [JsonProperty("name")]
    public string Name
    {
        get;
        set;
    }
}
```

Editing

You can edit the name, attributes, and methods of the class diagram shapes just double clicking, similar to editing a node annotation.



UML Activity diagram

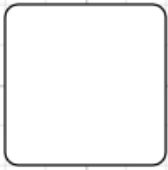
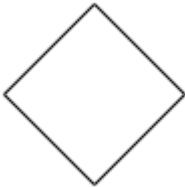
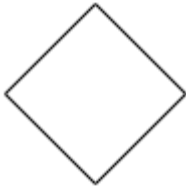



An Activity diagram functions as a visual flowchart, illustrating the progression from one activity to the next within a system. Each activity corresponds to a system operation, providing a clear depiction of the sequential flow in a dynamic process..


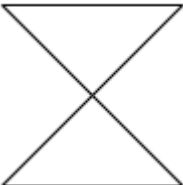
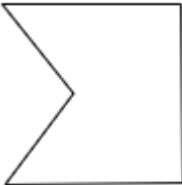
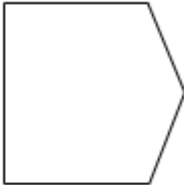
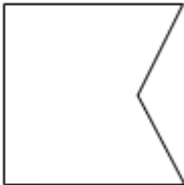
The purpose of an activity diagram can be described as follows.

1. Draw the activity flow of a system.
2. Describe the sequence from one activity to another.
3. Describe the parallel, branched, and concurrent flow of the system.

UML Activity diagram Shapes

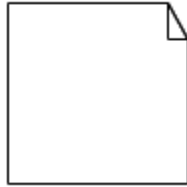
To create a UmlActivity, define type as "UmlActivity" and the list of built-in shapes as demonstrated as follows and it should be set in the "shape" property.

Shape	Image	
-----	-----	
Action		
Decision		
MergeNode		
InitialNode		
FinalNode		
ForkNode		

JoinNode	
TimeEvent	
AcceptingEvent	
SendSignal	
ReceiveSignal	



| StructuredNode |



| Note |

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

UMLACTIVITY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
```

```

        Width = 100,
        Shape = new {type="UmlActivity", shape="Action"}
    });
    ViewBag.nodes = Nodes;
    return View();
}
}
}

```

Uml Activity connector

To create an Uml Activity connector, define the type as "UmlActivity" and flow as either "Exception" or "Control" or "Object".

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

UMLACTIVITYCONNECTOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() {
                Id = "connector",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
                Shape = new {type = "UmlActivity", flow="Exception"}
            });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

Connector in Diagram

Connectors are objects used to create link between two points, nodes or ports to represent the relationships between them.

Create connector

Connector can be created by defining the source and target point of the connector. The path to be drawn can be defined with a collection of segments. To explore the properties of a [connector](#), refer to [Connector Properties](#).

Add connectors through connectors collection

The [sourcePoint](#) and [targetPoint](#) properties of connector allows to define the end points of a connector.

DEFAULT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 } });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}
```

Add connector at runtime

Connectors can be added at runtime by using public method, `diagram.add` and can be removed at runtime by using public method, `diagram.remove`.

DEFAULT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
```

```

        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 } });
        ViewBag.connectors = Connectors;
        return View();
    }
}
}

```

```
`javascript
```

```

var connectors = {
id: "connector2",
style: {
strokeColor: '#6BA5D7',
fill: '#6BA5D7',
strokeWidth: 2
},
targetDecorator: {
style: {
fill: '#6BA5D7',
strokeColor: '#6BA5D7'
}
},
sourcePoint: {
x: 300,
y: 100
},
targetPoint: {
x: 400,
y: 200
}
}

var diagram = document.getElementById("container").ej2_instances[0];
// Adds to the Diagram
diagram.add(connectors)

```



```
// Remove from the diagram
diagram.remove(connectors)
,
```

Connectors from palette

Connectors can be predefined and added to the symbol palette. You can drop those connectors into the diagram, when required.

For more information about adding connectors from symbol palette, refer to [Symbol Palette](#).

Draw connectors

Connectors can be interactively drawn by clicking and dragging on the diagram surface by using [drawingObject](#).

For more information about drawing connectors, refer to [Draw Connectors](#).

Update connector at runtime

Various connector properties such as [sourcePoint](#), [targetPoint](#), [style](#), [sourcePortID](#), [targetPortID](#), etc., can be updated at the runtime.

DEFAULT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 } });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}
```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];

// Update the connector properties at the run time
diagram.connectors[0].style.strokeColor = '#6BA5D7';
diagram.connectors[0].style.fill = '#6BA5D7';
diagram.connectors[0].style.strokeWidth = 2;
```

```

diagram.connectors[0].targetDecorator.style.fill = '#6BA5D7';
diagram.connectors[0].targetDecorator.style.strokeColor = '#6BA5D7';
diagram.connectors[0].sourcePoint.x = 150;
diagram.connectors[0].targetPoint.x = 150;
diagram.dataBind();

```

Connect nodes

- The [sourceID](#) and [targetID](#) properties allow to define the nodes to be connected.

CONNECT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "node2", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            nodes.Add(new DiagramNode() {
                Id = "node2",
                Width = 100,

```

```

        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 300,
        OffsetY = 300,
        Annotations = Node2
    });
    ViewBag.nodes = nodes;
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID="node1", TargetID="node2" });
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

- When you remove NodeConstraints InConnect from Default, the node accepts only an outgoing connection to dock in it. Similarly, when you remove NodeConstraints OutConnect from Default, the node accepts only an incoming connection to dock in it.
- When you remove both InConnect and OutConnect NodeConstraints from Default, the node restricts connector to establish connection in it.

```
`javascript
```

```
node.constraints = NodeConstraints.Default & ~NodeConstraints.InConnect,
```

```
,
```

Connections with ports

The [sourcePortID](#) and [targetPortID](#) properties allow to create connections between some specific points of source/target nodes.

PORTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            }

```

```

    });
    List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
    Node2.Add(new DiagramNodeAnnotation() {
        Content = "node2", Style = new DiagramTextStyle() {
            Color = "White", StrokeColor = "None"
        }
    });
    List<DiagramPort> ports1 = new List<DiagramPort>();
    ports1.Add(new CustomPort() { Id = "port1", Shape =
PortShapes.Circle, Offset = new DiagramPoint() { X = 1, Y = 0.5 },
Visibility = PortVisibility.Visible });
    List<DiagramPort> ports2 = new List<DiagramPort>();
    ports2.Add(new CustomPort() { Id = "port2", Shape =
PortShapes.Circle, Offset = new DiagramPoint() { X = 0, Y = 0.5 },
Visibility = PortVisibility.Visible });
    nodes.Add(new DiagramNode() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1,
        Ports = ports1
    });
    nodes.Add(new DiagramNode() {
        Id = "node2",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 300,
        OffsetY = 300,
        Annotations = Node2,
        Ports = port2
    });
    ViewBag.nodes = nodes;
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID="node1", TargetID="node2", SourcePortID = "port1", TargetPortID =
"port2" });
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

Similarly, the `sourcePortID` or `targetPortID` can be changed at the runtime by changing the port [sourcePortID](#) or [targetPortID](#).

PORTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "node2", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List<DiagramPort> ports1 = new List<DiagramPort>();
            ports1.Add(new CustomPort() { Id = "port1", Shape =
PortShapes.Circle, Offset = new DiagramPoint() { X = 1, Y = 0.5 },
Visibility = PortVisibility.Visible });
            List<DiagramPort> ports2 = new List<DiagramPort>();
            ports2.Add(new CustomPort() { Id = "port2", Shape =
PortShapes.Circle, Offset = new DiagramPoint() { X = 0, Y = 0.5 },
Visibility = PortVisibility.Visible });
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
                Ports = ports1
            });
            nodes.Add(new DiagramNode() {
                Id = "node2",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 300,
                OffsetY = 300,
                Annotations = Node2,
                Ports = port2
            });
        }
    }
}

```

```

        });
        ViewBag.nodes = nodes;
        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID="node1", TargetID="node2", SourcePortID = "port1", TargetPortID =
"port2" });
        ViewBag.connectors = Connectors;
        return View();
    }
}
}

```

`javascript

```

var diagram = document.getElementById("container").ej2_instances[0];
diagram.appendTo('#element');
// Update the target portID at the run time
diagram.connectors[0].targetPortID = 'newnodeport1'
`

```

- When you set PortConstraints to InConnect, the port accepts only an incoming connection to dock in it. Similarly, when you set PortConstraints to OutConnect, the port accepts only an outgoing connection to dock in it.
- When you set PortConstraints to None, the port restricts connector to establish connection in it.

`javascript

```

port.constraints = PortConstraints.InConnect,
`

```

Segments

The path of the connector is defined with a collection of segments. There are three types of segments.

Straight

To create a straight line, specify the [type](#) of the segment as **straight** and add a straight segment to [segments](#) collection and need to specify [type](#) for the connector.

CSHTML

```
@ (Html.EJS().Diagram("container").Width("100%").Height("580px"))
```

SEGMENT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;

```

```

using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 }, Type = Segments.Straight });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

The [point](#) property of straight segment allows to define the end point of it.

CSHTML

```
@(Html.EJS().Diagram("container").Width("100%).Height("580px"))
```

SEGMENT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 }, Type = Segments.Straight });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

Orthogonal

Orthogonal segments is used to create segments that are perpendicular to each other.

Set the segment [type](#) as orthogonal to create a default orthogonal segment and need to specify [type](#).

Multiple segments can be defined one after another. To create a connector with multiple segments, define and add the segments to [connector.segments](#) collection.

ORTHOGONAL.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 }, Type = Segments.Orthogonal });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

The [length](#) and [direction](#) properties allows to define the flow and length of segment.

DIRECTION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector", SourcePoint = new DiagramPoint() {
                    X = 100, Y = 100
                }, TargetPoint = new DiagramPoint() {
                    X = 300, Y = 300
                }, Type = Segments.Orthogonal, Segments = new Segments() {
                    Type = "Orthogonal",
                    Direction = "Bottom",
                    Length = 150
                }
            });
            ViewBag.connectors = Connectors;
            return View();
        }
        public class Segments {
            [DefaultValue(null)]

```



```

        [HtmlAttributeName("type")]
        [JsonProperty("type")]
        public string Type {
            get;
            set;
        }
        [DefaultValue(null)]
        [HtmlAttributeName("direction")]
        [JsonProperty("direction")]
        public string Direction {
            get;
            set;
        }
        [DefaultValue(null)]
        [HtmlAttributeName("length")]
        [JsonProperty("length")]
        public number Length {
            get;
            set;
        }
    }
}
}

```

Note: You need to mention the segment type as same as what you mentioned in connector type. There should be no contradiction between connector type and segment type.

Avoid overlapping

Orthogonal segments are automatically re-routed, in order to avoid overlapping with the source and target nodes.

OVERLAPPING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "node2", Style = new DiagramTextStyle() {

```

```

        Color = "White", StrokeColor = "None"
    }
    });
    List<DiagramPort> ports1 = new List<DiagramPort>();
    ports1.Add(new CustomPort() { Id = "port1", Shape =
PortShapes.Circle, Offset = new DiagramPoint() { X = 1, Y = 0.5 },
Visibility = PortVisibility.Visible });
    List<DiagramPort> ports2 = new List<DiagramPort>();
    ports2.Add(new CustomPort() { Id = "port2", Shape =
PortShapes.Circle, Offset = new DiagramPoint() { X = 0, Y = 0.5 },
Visibility = PortVisibility.Visible });
    nodes.Add(new DiagramNode() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1,
        Ports = ports1
    });
    nodes.Add(new DiagramNode() {
        Id = "node2",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 300,
        OffsetY = 300,
        Annotations = Node2,
        Ports = port2
    });
    ViewBag.nodes = nodes;
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID="node1", TargetID="node2", SourcePortID = "port1", TargetPortID =
"port2", Type = Segments.Orthogonal });
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

Bezier

Bezier segments are used to create curve segments and the curves are configurable either with the control points or with vectors.

To create a bezier segment, the [segment.type](#) is set as `bezier` and need to specify [type](#) for the connector.

VECTOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {

            List<vector> Stop = new List<vector>();
            Stop.Add(new vector() { Distance = 100, Angle = 90 });
            Stop.Add(new vector() { Distance = 90, Angle = 270 });
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector", SourcePoint = new DiagramPoint() {
                    X = 100, Y = 100
                }, TargetPoint = new DiagramPoint() {
                    X = 300, Y = 300
                }, Type = Segments.Besizer, Segments = new Segments() {
                    Type = "Besizer",
                    Vector = Stop
                }
            });
            ViewBag.connectors = Connectors;
            return View();
        }
        public class Segments {
            [DefaultValue(null)]
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type {
                get;
                set;
            }
            [DefaultValue(null)]
            [HtmlAttributeName("vectors")]
            [JsonProperty("vectors")]
            public List<vector> Vector {
                get;
                set;
            }
        }
        public class vector {
            [DefaultValue(null)]
            [HtmlAttributeName("distance")]
            [JsonProperty("distance")]
            public number Distance {
                get;
                set;
            }
            [DefaultValue(null)]
            [HtmlAttributeName("angle")]

```

```

        [JsonProperty("angle")]
        public number Angle {
            get;
            set;
        }
    }
}

```

The [point1](#) and [point2](#) properties of bezier segment enables to set the control points.

The [vector1](#) and [vector2](#) properties of bezier segment enable you to define the vectors.

POINTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {

            List<point> Stop = new List<vector>();
            Stop.Add(new point() { X = 100, Y = 100 });
            Stop.Add(new point() { X = 200, Y = 200 });
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector", SourcePoint = new DiagramPoint() {
                    X = 100, Y = 100
                }, TargetPoint = new DiagramPoint() {
                    X = 300, Y = 300
                }, Type = Segments.Besizer, Segments = new Segments() {
                    Type = "Besizer",
                    Vector = Stop
                }
            });
            ViewBag.connectors = Connectors;
            return View();
        }
        public class Segments {
            [DefaultValue(null)]
            [HtmlAttributeName("type")]
            [JsonProperty("type")]
            public string Type {
                get;
                set;
            }
            [DefaultValue(null)]
            [HtmlAttributeName("points")]
            [JsonProperty("points")]
            public List<point> Point {

```

```

        get;
        set;
    }
}

public class point {
    [DefaultValue(null)]
    [HtmlAttributeName("x")]
    [JsonProperty("x")]
    public double X {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("y")]
    [JsonProperty("y")]
    public double Y {
        get;
        set;
    }
}
}
}

```

Decorator

- Starting and ending points of a connector can be decorated with some customizable shapes like arrows, circles, diamond, or path. The connection end points can be decorated with the [sourceDecorator](#) and [targetDecorator](#) properties of the connector.
- The [shape](#) property of [sourceDecorator](#) allows to define the shape of the decorators. Similarly, the [shape](#) property of [targetDecorator](#) allows to define the shape of the decorators.
- To create custom shape for source decorator, use [pathData](#) property. Similarly, to create custom shape for target decorator, use [pathData](#) property.

SRCDECORATOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector", SourcePoint = new DiagramPoint() {
                    X = 100, Y = 100
                }, TargetPoint = new DiagramPoint() {
                    X = 300, Y = 300
                }
            });
        }
    }
}

```

```

        }, Type = Segments.Straight, SourceDecorator = new
DiagramDecorator() {
    Shape = DecoratorShapes.Circle, Style = new
DiagramShapeStyle() {
        StrokeColor = "Red", StrokeWidth = 3
    }
    }, TargetDecorator = new DiagramDecorator() {
        Shape = DecoratorShapes.Custom,
        PathData = "M80.5,12.5 C80.5,19.127417 62.59139,24.5
40.5,24.5 C18.40861,24.5 0.5,19.127417 0.5,12.5 C0.5,5.872583 18.40861,0.5
40.5,0.5 C62.59139,0.5 80.5,5.872583 80.5,12.5 z",
        Style = new DiagramShapeStyle() {
            StrokeColor = "Red", StrokeWidth = 3, opacity = 0.8
        }
    }
    });
Connectors.Add(new DiagramConnector() {
    Id = "connector", SourcePoint = new DiagramPoint() {
        X = 400, Y = 100
    }, TargetPoint = new DiagramPoint() {
        X = 600, Y = 300
    }, Type = Segments.Straight, SourceDecorator = new
DiagramDecorator() {
        Shape = DecoratorShapes.IndentedArrow, Style = new
DiagramShapeStyle() {
            StrokeColor = "Blue", StrokeWidth = 3
        }
    },
    TargetDecorator = new DiagramDecorator() {
        Shape = DecoratorShapes.OutdentedArrow, Style = new
DiagramShapeStyle() {
            StrokeColor = "Yellow", StrokeWidth = 3
        }
    }
    });
ViewBag.connectors = Connectors;
return View();
}
}
}

```

Padding

Padding is used to leave the space between the Connector's end point and the object to where it is connected.

- The [sourcePadding](#) property of connector defines space between the source point and the source node of the connector.
- The [targetPadding](#) property of connector defines space between the end point and the target node of the connector.

PADDING.CS

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
            });
            nodes.Add(new DiagramNode() {
                Id = "node2",
                Width = 100,
                Height = 100,
                OffsetX = 300,
                OffsetY = 300,
            });
            ViewBag.nodes = nodes;
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID="node1", TargetID="node2",sourcePadding =20,
targetPadding =20 });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

Hit padding

The [hitPadding](#) property enables you to define the clickable area around the connector path. The default value for hitPadding is 10.

PADDING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {

```

```

        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 },hitPadding =20,
        });
        ViewBag.connectors = Connectors;
        return View();
    }
}

```

Flip

The diagram Provides support to flip the connector. The [flip](#) is performed to give the mirrored image of the original element.

The flip types are as follows:

- HorizontalFlip - Horizontal is used to interchange the connector source and target x points.
- VerticalFlip - Vertical is used to interchange the connector source and target y points.
- Both - Both is used to interchange the source point as target point and target point as source point.

FLIP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector", SourcePoint = new DiagramPoint() {
                    X = 100, Y = 100
                }, TargetPoint = new DiagramPoint() {
                    X = 300, Y = 300
                },flip = FlipDirection.Horizontal,
                TargetDecorator = new DiagramDecorator() {
                    Shape = DecoratorShapes.OutdentedArrow, Style = new
DiagramShapeStyle() {
                        StrokeColor = "Yellow", StrokeWidth = 3
                    }
                }
            });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```



```
}
}
```

Note: The flip is not applicable when the connectors connect in nodes.

Bridging

Line bridging creates a bridge for lines to smartly cross over the other lines, at points of intersection. By default, [bridgeDirection](#) is set to top. Depending upon the direction given bridging direction appears. Bridging can be enabled or disabled either with the `connector.constraints` or `diagram.constraints`.

BRIDGING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "node2", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 150,
                Height = 60,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 300,
                OffsetY = 60,
                Annotations = Node1
            });
            nodes.Add(new DiagramNode() {
                Id = "node2",
                Width = 150,
                Height = 60,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
            },
```

```

        OffsetX = 300,
        OffsetY = 250,
        Annotations = Node2
    });
    ViewBag.nodes = nodes;
    List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
    Connectors.Add(new DiagramConnector() {
        Id = "connector", SourceID = "node1", TargetID = "node2",
Type = Segments.Straight
    });
    Connectors.Add(new DiagramConnector() {
        Id = "connector", SourcePoint = new DiagramPoint() {
            X = 200, Y = 130
        }, TargetPoint = new DiagramPoint() {
            X = 400, Y = 130
        }, Type = Segments.Straight
    });
    Connectors.Add(new DiagramConnector() {
        Id = "connector", SourcePoint = new DiagramPoint() {
            X = 200, Y = 170
        }, TargetPoint = new DiagramPoint() {
            X = 400, Y = 170
        }, Type = Segments.Straight
    });
    ViewBag.connectors = Connectors;
    return View();
}
}

```

Note: You need to inject connector bridging module into the diagram.

The [bridgeSpace](#) property of connectors can be used to define the width for line bridging.

Limitation: Bezier segments do not support bridging.

Corner radius

Corner radius allows to create connectors with rounded corners. The radius of the rounded corner is set with the [cornerRadius](#) property.

CORNERRADIUS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();

```

```

        Node1.Add(new DiagramNodeAnnotation() {
            Content = "node1", Style = new DiagramTextStyle() {
                Color = "White", StrokeColor = "None"
            }
        });
        List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
        Node2.Add(new DiagramNodeAnnotation() {
            Content = "node2", Style = new DiagramTextStyle() {
                Color = "White", StrokeColor = "None"
            }
        });
        nodes.Add(new DiagramNode() {
            Id = "node1",
            Width = 100,
            Height = 100,
            Style = new NodeStyleNodes() {
                Fill = "darkcyan"
            },
            OffsetX = 100,
            OffsetY = 100,
            Annotations = Node1
        });
        nodes.Add(new DiagramNode() {
            Id = "node2",
            Width = 100,
            Height = 100,
            Style = new NodeStyleNodes() {
                Fill = "darkcyan"
            },
            OffsetX = 300,
            OffsetY = 300,
            Annotations = Node2
        });
        ViewBag.nodes = nodes;
        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() { Id = "connector",
SourceID="node1", TargetID="node2", CornerRadius= 10, Type =
Segments.Orthogonal });
        ViewBag.connectors = Connectors;
        return View();
    }
}

```

Appearance

- The connector's [strokeWidth](#), [strokeColor](#), [strokeDashArray](#), and [opacity](#) properties are used to customize the appearance of the connector segments.
- The [visible](#) property of the connector enables or disables the visibility of connector.
- Default values for all the connectors can be set using the [getConnectorDefaults](#) properties. For example, if all connectors have the same type or having the same property then such properties can be moved into [getConnectorDefaults](#).

Segment appearance

STYLE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 }, Type = Segments.Orthogonal, Style = new
DiagramStrokeStyle() { StrokeDashArray = "2,2", StrokeColor = "Red",
StrokeWidth = 2 }});
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

Decorator appearance

- The source decorator's [strokeColor](#), [strokeWidth](#), and [strokeDashArray](#) properties are used to customize the color, width, and appearance of the decorator.
- To set the border stroke color, stroke width, and stroke dash array for the target decorator, use [strokeColor](#), [strokeWidth](#), and [strokeDashArray](#).
- To set the size for source and target decorator, use width and height property.

DECORATOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector", SourcePoint = new DiagramPoint() {
                    X = 100, Y = 100
                }, TargetPoint = new DiagramPoint() {

```

```

        X = 300, Y = 300
    }, Type = Segments.Orthogonal, SourceDecorator = new
DiagramDecorator() {
    Shape = DecoratorShapes.IndentedArrow, Style = new
DiagramShapeStyle() {
    StrokeColor = "Blue", StrokeWidth = 3
    }
},
    TargetDecorator = new DiagramDecorator() {
    Shape = DecoratorShapes.OutdentedArrow, Style = new
DiagramShapeStyle() {
    StrokeColor = "Yellow", StrokeWidth = 3
    }
    }
});
ViewBag.connectors = Connectors;
return View();
}
}
}

```

Interaction

- Diagram allows to edit the connectors at runtime. To edit the connector segments at runtime, refer to [Connection Editing](#).

Automatic line routing

Diagram provides additional flexibility to re-route the diagram connectors. A connector will frequently re-route itself when a shape moves next to it. The following screenshot illustrates how the connector automatically re-routes the segments.

- The line routing constraints must be included to the default diagram constraints to enable automatic line routing support like below.
- The following code block shows how to create the diagram with specifying nodes, connectors, constraints, and necessary modules for line routing.

CSHTML

```

@ (Html.EJS().Diagram("container").Width("100%").Height("580px")
    .Constraints(DiagramConstraints.Default |
DiagramConstraints.LineRouting)

```

LINEROUTING.CS

```

using System.Collections.Generic;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        // GET: Connector
    }
}

```

```

public ActionResult Connector()
{
    List<DiagramConnector> connectors = new
List<DiagramConnector>();
    List<DiagramConnectorAnnotation> annotations = new
List<DiagramConnectorAnnotation>();
    annotations.Add(new DiagramConnectorAnnotation() { Offset = .7,
Content = "Routing \n enabled" });
    connectors.Add(new DiagramConnector() { Id = "connector",
SourceID = "shape1", TargetID = "node2", Type = Segments.Orthogonal,
Annotations = annotations });
    annotations = new List<DiagramConnectorAnnotation>();
    annotations.Add(new DiagramConnectorAnnotation() { Offset = .7,
Content = "Routing \n disabled" });
    connectors.Add(new DiagramConnector() { Id = "connector2",
SourceID = "shape1", TargetID = "node2", Type = Segments.Orthogonal,
Annotations = annotations });
    List<Syncfusion.EJ2.Diagrams.DiagramNode> nodes = new
List<Syncfusion.EJ2.Diagrams.DiagramNode>();
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 100, OffsetY = 100,
Height = 50, Width = 120 });
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 300, OffsetY = 300,
Height = 50, Width = 120 });
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 150, OffsetY = 200,
Height = 50, Width = 120 });
    ViewBag.nodes = nodes;
    ViewBag.connectors = connectors;
    return View();
}
}
}

```

- In some situations, automatic line routing enabled diagram needs to ignore a specific connector from automatic line routing. So, in this case, auto routing feature can be disabled to the specific connector using the [constraints](#) property of the connector.

CSHTML

```

@ (Html.EJS().Diagram("container").Width("100%").Height("580px")
.Constraints(DiagramConstraints.Default |
DiagramConstraints.LineRouting)

```

LINEROUTING1.CS

```

using System.Collections.Generic;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
    }
}

```

```

// GET: Connector
public ActionResult Connector()
{
    List<DiagramConnector> connectors = new
List<DiagramConnector>();
    List<DiagramConnectorAnnotation> annotations = new
List<DiagramConnectorAnnotation>();
    annotations.Add(new DiagramConnectorAnnotation() { Offset = .7,
Content = "Routing \n enabled" });
    connectors.Add(new DiagramConnector() { Id = "connector",
SourceID = "shape1", TargetID = "node2", Type = Segments.Orthogonal,
Annotations = annotations });
    annotations = new List<DiagramConnectorAnnotation>();
    annotations.Add(new DiagramConnectorAnnotation() { Offset = .7,
Content = "Routing \n disabled" });
    connectors.Add(new DiagramConnector() { Id = "connector2",
SourceID = "shape1", TargetID = "node2", Constraints =
ConnectorConstraints.Default & ~ConnectorConstraints.InheritLineRouting
Type = Segments.Orthogonal, Annotations = annotations });
    List<Syncfusion.EJ2.Diagrams.DiagramNode> nodes = new
List<Syncfusion.EJ2.Diagrams.DiagramNode>();
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 100, OffsetY = 100,
Height = 50, Width = 120 });
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 350, OffsetY = 350,
Height = 50, Width = 120 });
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 150, OffsetY = 200,
Height = 50, Width = 120 });
    List<DiagramNode> Node1 = new List<DiagramNode>();
    nodes.Add(new DiagramNode() { OffsetX = 300, OffsetY = 200,
Height = 50, Width = 120 });
    ViewBag.nodes = nodes;
    ViewBag.connectors = connectors;
    return View();
}
}
}

```

Constraints

- The [constraints](#) property of connector allows to enable or disable certain features of connectors.
- To enable or disable the constraints, refer [constraints](#).

CONSTRAINTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {

```

```

public partial class DiagramController: Controller {
    // GET: Nodes
    public ActionResult Nodes() {
        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 }, Constraints =
ConnectorConstraints.Default & ~ConnectorConstraints.Select });
        ViewBag.connectors = Connectors;
        return View();
    }
}

```

Custom properties

- The [addInfo](#) property of connectors allows to maintain additional information to the connectors.

```

`javascript
var connectors = {
id: 'connector1',
// Defines the information about the connector
addInfo: 'centralconnector',
type: 'Straight',
sourceID: 'Transaction',
targetID: 'Verification'
};

```

Stack order

The connectors [zIndex](#) property specifies the stack order of the connector. A connector with greater stack order is always in front of a connector with a lower stack order.

ZINDEX.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();

```



```

        Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 }, ZIndex = 1 });
        ViewBag.connectors = Connectors;
        return View();
    }
}
}

```

See Also

- [How to add annotations to the connector](#)
- [How to enable/disable the behavior of the node](#)
- [How to add connectors to the symbol palette](#)
- [How to perform the interaction on the connector](#)
- [How to create diagram connectors using drawing tools](#)

Group in Diagram Control

Group is used to cluster multiple nodes and connectors into a single element. It acts like a container for its children (nodes, groups, and connectors). Every change made to the group also affects the children. Child elements can be edited individually.

Create group

Add group when initializing diagram

A group can be added to the diagram model through [nodes](#) collection. To define an object as group, add the child objects to the [children](#) collection of the group.

- The [padding](#) property of a group node defines the spacing between the group node's edges and its children.
- While creating group, its child node needs to be declared before the group declaration.
- Add a node to the existing group child by using the `diagram.group` method.
- The group's `diagram.unGroup` method is used to define whether the group can be ungrouped or not.
- A group can be added into a child of another group.

GROUP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();

```

```

        Node1.Add(new DiagramNodeAnnotation() {
            Content = "node1", Style = new DiagramTextStyle() {
                Color = "White", StrokeColor = "None"
            }
        });
        List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
        Node2.Add(new DiagramNodeAnnotation() {
            Content = "Node2"
        });
        List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
        Node3.Add(new DiagramNodeAnnotation() {
            Content = "Node3"
        });
        nodes.Add(new Node() {
            Id = "node1",
            Width = 100,
            Height = 100,
            Style = new NodeStyleNodes() {
                fill = "#6BA5D7",
                strokeColor = "White"
            },
            text = "node1",
            OffsetX = 100,
            OffsetY = 100,
            Annotations = Node1,
        });
        nodes.Add(new Node() {
            Id = "node2", OffsetX = 100, OffsetY = 170, Annotations =
Node2
        });
        nodes.Add(new Node() {
            Id = "node3", OffsetX = 100, OffsetY = 240, Annotations =
Node3
        });
        string[] children = {"node1", "node2"};
        nodes.Add(new Node() {
            Id = "group", Children = children
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
diagram.selectAll();
```

```
// Adding the third node into the existing group
```

```
diagram.group();
```

```
,
```

UNGROUP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Annotations =
Node2
            });
            string[] children = {"node1", "node2"};
            nodes.Add(new Node() {
                Id = "group", Children = children
            });
            ViewBag.nodes = nodes;
```

```

        return View();
    }
}
public class Node: DiagramNode {
    public string text;
}
}

```

`javascript

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
diagram.selectAll();
```

```
// Ungroup the selected group into nodes
```

```
diagram.unGroup();
```

```
,
```

Add group at runtime

A group node can be added at runtime by using the client-side method `diagram.add`.

RUN.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,

```

```

        Style = new NodeStyleNodes() {
            fill = "#6BA5D7",
            strokeColor = "White"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1,
    });
    nodes.Add(new Node() {
        Id = "node2", OffsetX = 100, OffsetY = 170, Annotations =
Node2
    });
    nodes.Add(new Node() {
        Id = "node3", OffsetX = 100, OffsetY = 240, Annotations =
Node3
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
var group= {
```

```
id: 'group2',
```

```
children: ['node1', 'node2']
```

```
};
```

```
// Add the group into the diagram
```

```
diagram.add(group);
```

```
,
```

Add children to group at runtime

A childNode can be added to the specified Group at runtime by utilizing the client-side method `diagram.addChildToGroup`.

This functionality is achieved by passing the group and existing children as arguments to the method.

The following code illustrates how a child node and a group node can be passed as arguments to the method and executed at runtime.

```
`html
```

```
diagram.addChildToGroup(groupNode, childNode);
```

```
,
```

Remove children from group at runtime

A specific child from a group node can be removed at runtime by utilizing the client-side method `diagram.removeChildFromGroup`.

This functionality is achieved by passing the group and its children as arguments to the method.

The following code illustrates how a child node is removed from a group at runtime.

```
`html
```

```
diagram.removeChildFromGroup (groupNode, childNode);
```

```
`
```

GROUPCHILD.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
        }
    }
}
```

```

        nodes.Add(new Node() {
            Id = "node2", OffsetX = 100, OffsetY = 170, Annotations =
Node2
        });
        string[] children = {"node1", "node2"};
        nodes.Add(new Node() {
            Id = "group", Children = children
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
//Assign the group Node
```

```
let group=diagram.nodes[3];
```

```
//Assign the child Node
```

```
let child=diagram.nodes[2];
```

```
//To Add child to specific group at Runtime
```

```
diagramInstance.addChildToGroup(group, child);
```

```
//To remove the specific children from group at runtime
```

```
diagramInstance.removeChildFromGroup(group, child);
```

```
`
```

Container

Containers are used to automatically measure and arrange the size and position of the child elements in a predefined manner. There are two types of containers available.

Canvas

- The canvas panel supports absolute positioning and provides the least layout functionality to its contained diagram elements.
- Canvas allows to position its contained elements by using the margin and alignment properties.
- Rendering alone possible in canvas container.
- It allows elements to be either vertically or horizontally aligned.
- Child can be defined with the collection `canvas.children` property.
- Basic element can be defined with the collection of `basicElements`.

Stack

- Stack panel is used to arrange its children in a single line or stack order, either vertically or horizontally.
- It controls spacing by setting margin properties of child and padding properties of group. By default, a stack panel's [orientation](#) is vertical.

STACK.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.setNodeTemplate = "setNodeTemplate";
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

```
`javascript
```

```
function getTextElement(text) {
```



```
let textElement: TextElement = new TextElement();
textElement.width = 50;
textElement.height = 20;
textElement.content = text;
return textElement;
}

function addRows (column) {
column.children.push(getTextElement('Row1'));
column.children.push(getTextElement('Row2'));
column.children.push(getTextElement('Row3'));
column.children.push(getTextElement('Row4'));
}

function getNodeDefaults (node) {
node.height = 100;
node.width = 100;
node.style.fill = '#6BA5D7';
node.style.strokeColor = 'white';
return node;
},

function setNodeTemplate(obj, diagram) {
if (obj.id.indexOf('node5') !== -1) {
// It will be replaced with grid panel
var table = new StackPanel();
table.orientation = 'Horizontal';
table.padding.left
var column1 = new StackPanel();
column1.children = [];
column1.children.push(getTextElement('Column1'));
addRows(column1);
var column2 = new StackPanel();
column2.children = [];
column2.children.push(getTextElement('Column2'));
addRows(column2);
```

```

table.children = [column1, column2];
return table;
}
return null;
}
`

```

Difference between a basic group and containers

Group	Container
-----	-----
It arranges the child elements based on the child elements position and size properties.	Each container has a predefined behavior to measure and arrange its child elements. Canvas and stack containers are supported in the diagram.
The Padding, Min, and Max Size properties are not applicable for basic group.	It is applicable for container.
The Children's margin and alignment properties are not applicable for basic group.	It is applicable for container.

Interaction

You can edit the group and its children at runtime. For more information about how to interact with a group, refer to [Edit Groups](#).

See Also

- [How to add annotations to the node](#)
- [How to add ports to the node](#)
- [How to enable/disable the behavior of the node](#)
- [How to add nodes to the symbol palette](#)
- [How to create diagram nodes using drawing tools](#)
- [How to perform the interaction on the group](#)

Swimlane

Swimlane is a type of diagram nodes, which is typically used to visualize the relationship between a business process and the department responsible for it by focusing on the logical relationships between activities.

Create a swimlane

To create a swimlane, the type of shape should be set as [swimlane](#). By Default swimlane's are arranged vertically.

Headers

Header was the primary element for swimlanes. The [header](#) property of swimlane allows to define its textual description and to customize its appearance.

Note: By using this header, the swimlane interaction will be performed, like selection, dragging, etc.

SWIMLANEHEADER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List<Lane> Lanes = new List<Lane>();
            List<Phase> Phases = new List<Phase>();
            swimlane.Shape = new SwimLane() {
                Type = "SwimLane",
                PhaseSize = 20,
                Header = new Header()
                {
                    Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
                    Height = 50,
                    Orientation = "Horizontal",
                    Style = new DiagramTextStyle() { FontSize = 11 }
                },
                Lanes = Lanes,
                Phases = Phases
            };
            nodes.Add(swimlane);
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Customization of headers

The height and width of swimlane header can be customized with [weight](#) and [height](#) properties of swimlane header. Set fill color of header by using the [style](#) property. The orientation of swimlane can be customized with the [orientation](#) property of the header.

Note: By default the swimlane orientation has Horizontal.

HEADCUSTOMIZE.CS

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List<Lane> Lanes = new List<Lane>();
            List<Phase> Phases = new List<Phase>();
            swimlane.Shape = new SwimLane() {
                Type = "SwimLane",
                PhaseSize = 20,
                Header = new Header()
                {
                    Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
                    Height = 50,
                    Orientation = "Horizontal",
                    Style = new DiagramTextStyle() { FontSize = 11 ,fill
="red"}
                },
                Lanes = Lanes,
                Phases = Phases
            };
            nodes.Add(swimlane);
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Dynamic customization of swimlane header

You can customize the swimlane header style and text properties dynamically.

`typescript

```
let lane : nodeModel = diagram.nodes[0];
```

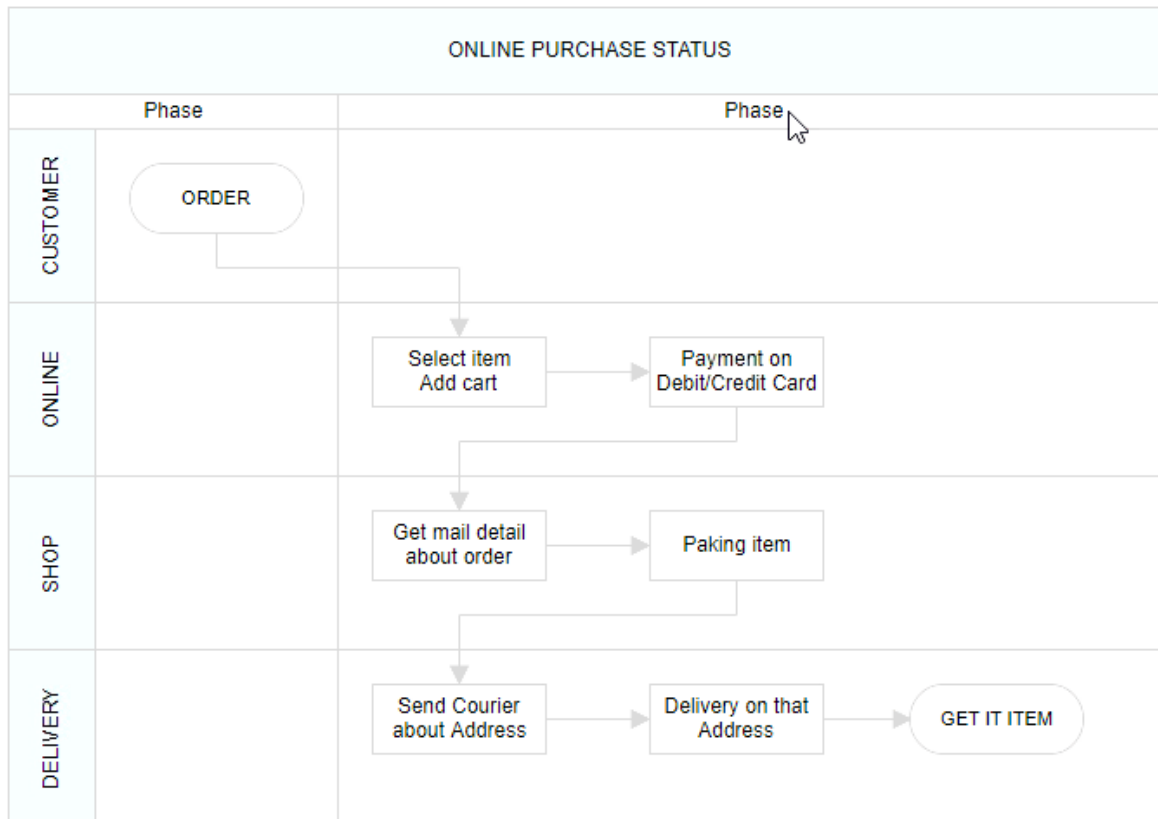
```
lane.shape.lanes[0].header.style.fill ='red';
```

```
diagram.dataBind();
```

`

Header editing

Diagram provides the support to edit swimlane headers at runtime. The header editing is achieved by double click event. Double clicking the header label will enable the editing.



Lanes

Lane is a functional unit or a responsible department of a business process that helps to map a process within the functional unit or in between other functional units.

The number of [lanes](#) can be added to swimlane. The lanes are automatically stacked inside swimlane based on the order they are added.

Create an empty lane

- The lanes `id` is used to define the name of the lane and its further used to find the lane at runtime and do any customization.
- The additional information to the lane is provided by using the [addInfo](#) property of the lane.

EMPTYLANE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {

```

```

public partial class DiagramController: Controller {
    // GET: Nodes
    public ActionResult Nodes() {
        List < DiagramNode > nodes = new List < DiagramNode > ();
        List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
        Node1.Add(new DiagramNodeAnnotation() {
            Content = "node1", Style = new DiagramTextStyle() {
                Color = "White", StrokeColor = "None"
            }
        });
        List<Lane> Lanes = new List<Lane>();
        List<Phase> Phases = new List<Phase>();
        swimlane.Shape = new SwimLane(){
            Type = "SwimLane",
            PhaseSize = 20,
            Header = new Header()
            {
                Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
                Height = 50,
                Orientation = "Horizontal",
                Style = new DiagramTextStyle() { FontSize = 11 ,fill
="red"}
            },
            Lanes = Lanes,
            Phases = Phases
        };
        nodes.Add(swimlane);
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

Create lane header

- The [header](#) property of lane allows to textually describe the lane and to customize the appearance of the description.

LANEHEADER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes

```

```

    public ActionResult Nodes() {
        List < DiagramNode > nodes = new List < DiagramNode > ();
        List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
        Node1.Add(new DiagramNodeAnnotation() {
            Content = "node1", Style = new DiagramTextStyle() {
                Color = "White", StrokeColor = "None"
            }
        });
        List<Lane> Lanes = new List<Lane>();
        Lanes.Add(new Lane()
        {
            Id = "stackCanvas1",
            Height = 100,
            Header = new Header()
            {
                Annotation = new DiagramNodeAnnotation() { Content =
"Consumer" },
                Width = 50,
            },
        });
        List<Phase> Phases = new List<Phase>();
        swimlane.Shape = new SwimLane() {
            Type = "SwimLane",
            PhaseSize = 20,
            Header = new Header()
            {
                Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
                Height = 50,
                Orientation = "Horizontal",
                Style = new DiagramTextStyle() { FontSize = 11 ,fill
="red"}
            },
            Lanes = Lanes,
            Phases = Phases
        };
        nodes.Add(swimlane);
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

Customizing lane header

- The size of lane can be controlled by using the [width](#) and [height](#) properties of the lane.
- The appearance of the lane can be set by using the [style](#) properties.

LANEHEADERCUSTOMIZE.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List<Lane> Lanes = new List<Lane>();
            Lanes.Add(new Lane()
            {
                Id = "stackCanvas1",
                Height = 100,
                Header = new Header()
                {
                    Annotation = new DiagramNodeAnnotation() { Content =
"Consumer" },
                    Width = 50,Style = new DiagramTextStyle() { FontSize =
11 ,fill ="red"}
                },
            });
            List<Phase> Phases = new List<Phase>();
            swimlane.Shape = new SwimLane(){
                Type = "SwimLane",
                PhaseSize = 20,
                Header = new Header()
                {
                    Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
                    Height = 50,
                    Orientation = "Horizontal",
                    Style = new DiagramTextStyle() { FontSize = 11 ,fill
="red"}
                },
                Lanes = Lanes,
                Phases = Phases
            };
            nodes.Add(swimlane);
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```


Dynamic customization of lane header

You can customize the lane header style and text properties dynamically.

```
`typescript
let lane : nodeModel = diagram.nodes[0];
lane.shape.lanes[0].header.style.fill = 'red';
diagram.dataBind();
`
```

Add lane at runtime

You can add the lane at runtime by using the client side API method called `addLanes`.

Add children to lane

To add nodes to lane, you should add [children](#) collection of the lane.

CSHTML

```
<div class="col-lg-12 control-section">
  <div class="content-wrapper">
    @(Html.EJS().Diagram("container")
      .Width("100%")
      .Height("500px")
      .Nodes(ViewBag.nodes).Render())
  </div>
</div>
```

LANECHILDREN.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> Nodes = new List<DiagramNode>();
            //Create lanes
            List<Lane> Lanes = new List<Lane>();
            Lanes.Add(new Lane()
            {
                Id = "stackCanvas1",
                Height = 100,
                Header = new Header()
                {
                    Annotation = new DiagramNodeAnnotation() { Content =
"Consumer" },
                    Width = 50,
                },
            },
```

```

        Children = firstLaneChildren
    });
    Lanes.Add(new Lane()
    {
        Id = "stackCanvas2",
        Height = 100,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"Marketing" },
            Width = 50,
        },
        Children = secondLaneChildren
    });
    Lanes.Add(new Lane()
    {
        Id = "stackCanvas3",
        Height = 100,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"Sales" },
            Width = 50,
        },
        Children = thirdLaneChildren
    });
    Lanes.Add(new Lane()
    {
        Id = "stackCanvas4",
        Height = 100,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"Success" },
            Width = 50,
        },
        Children = fourthLaneChildren
    });
    //Create phases
    List<Phase> Phases = new List<Phase>();
    Phases.Add(new Phase()
    {
        Id = "phase1",
        Offset = 170,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"Phase" },
        },
    });
    swimlane.Shape = new SwimLane()
    {
        Type = "SwimLane",
        PhaseSize = 20,
        Header = new Header()
        {

```

```

        Annotation = new DiagramNodeAnnotation() { Content =
"SALES PROCESS FLOW CHART" },
        Height = 50,
        Orientation = "Horizontal",
        Style = new DiagramTextStyle() { FontSize = 11 }
    },
    Lanes = Lanes,
    Phases = Phases
};
Nodes.Add(swimlane);
return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

Dynamically add children to lane

The diagram provides support to add children dynamically.

`typescript

```

let node = {
id: 'Order',
shape: { type: 'Path', data: pathData },
annotations: [
{
content: 'ORDER',
style: { fontSize: 11 }
}],
margin: { left: 60, top: 20 },
height: 40, width: 100
}
let lane : nodeModel = diagram.nodes[0];
swimlane.shape.lanes[0].children[0]=node;
diagram.dataBind();
`

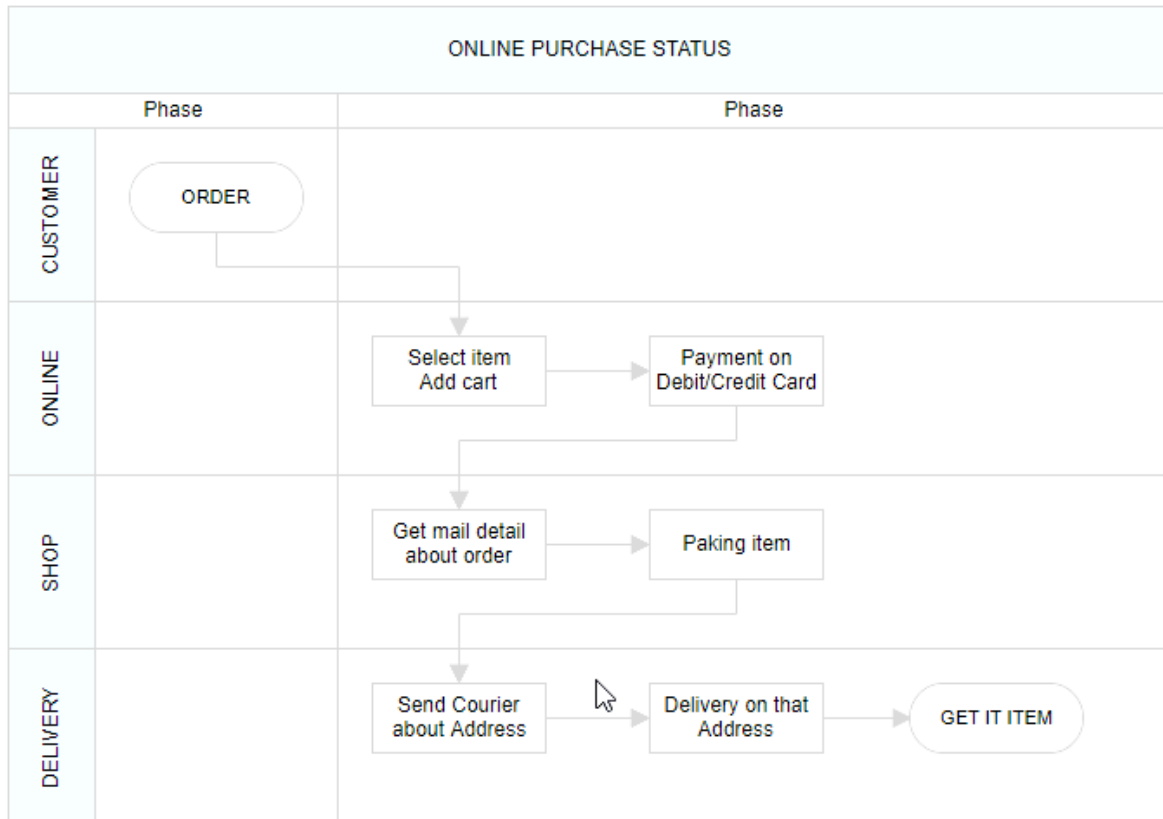
```

Lane interaction

Resizing lane

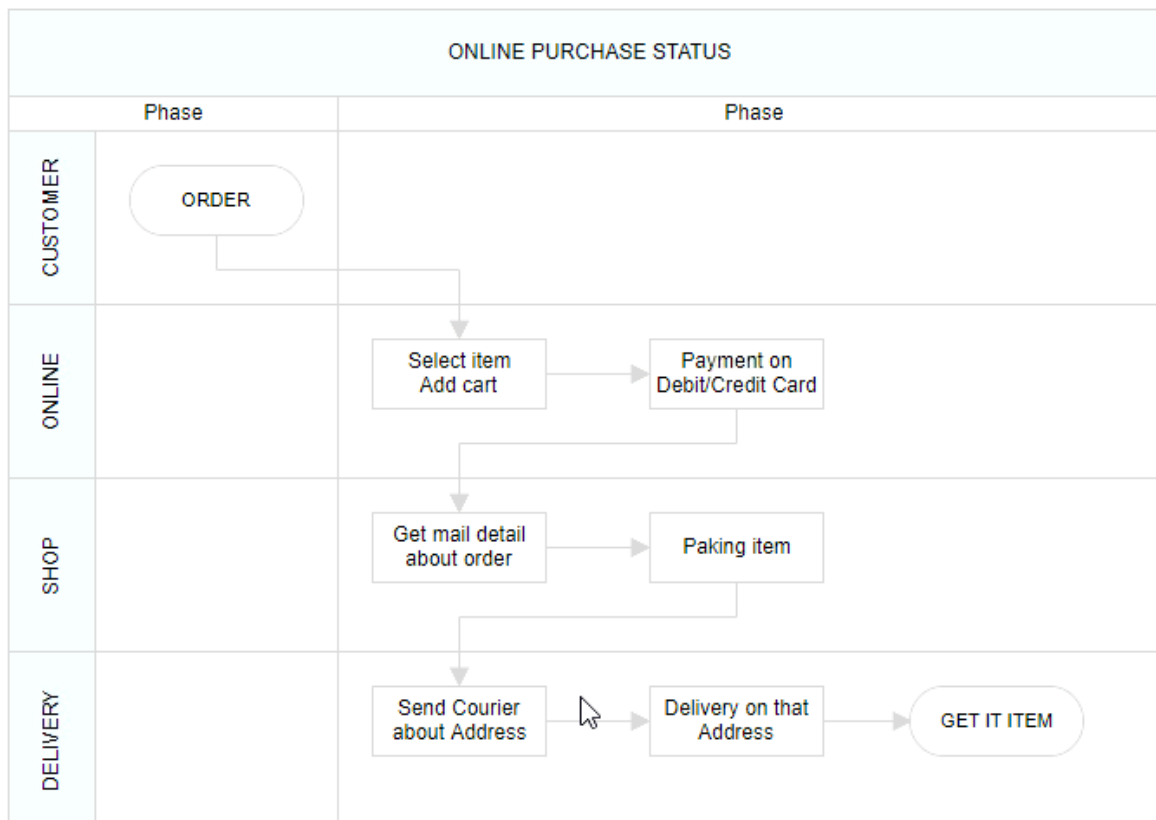
- Lane can be resized in the bottom and left direction.
- Lane can be resized by using resize selector of the lane.
- Once you can resize the lane, the swimlane will be resized automatically.

- The lane can be resized by either resizing the selector or the tight bounds of the child object. If the child node moves to the edge of the lane, it can be automatically resized.



Lane swapping

- Lanes can be swapped using drag the lanes over another lane.
- Helper should intimate the insertion point while lane swapping.



Disable Swimlane Lane swapping

You can disable swimlane lane swapping by using the property called `canMove`.

``typescript`

```
let node = {
  id: 'Order',
  shape: { type: 'Path', data: pathData },
  annotations: [
    {
      content: 'ORDER',
      style: { fontSize: 11 }
    }
  ],
  margin: { left: 60, top: 20 },
  height: 40, width: 100, canMove: false
}

let lane : nodeModel = diagram.nodes[0];
swimlane.shape.lanes[0].children[0]=node;
diagram.dataBind();
```

Resize helper

- The special resize helper will be used to resize the lanes.
- The resize cursor will be available on the left and bottom direction alone.
- Once the lane is resized, the swimlane will be resized automatically.

Children interaction in lanes

- You can resize the child node within swimlanes.
- You can drag the child nodes within lane.
- Interchange the child nodes from one lane to another lane.
- Drag and drop the child nodes from lane to diagram.
- Drag and drop the child nodes from diagram to lane.
- Based on the child node interactions, the lane size should be updated.

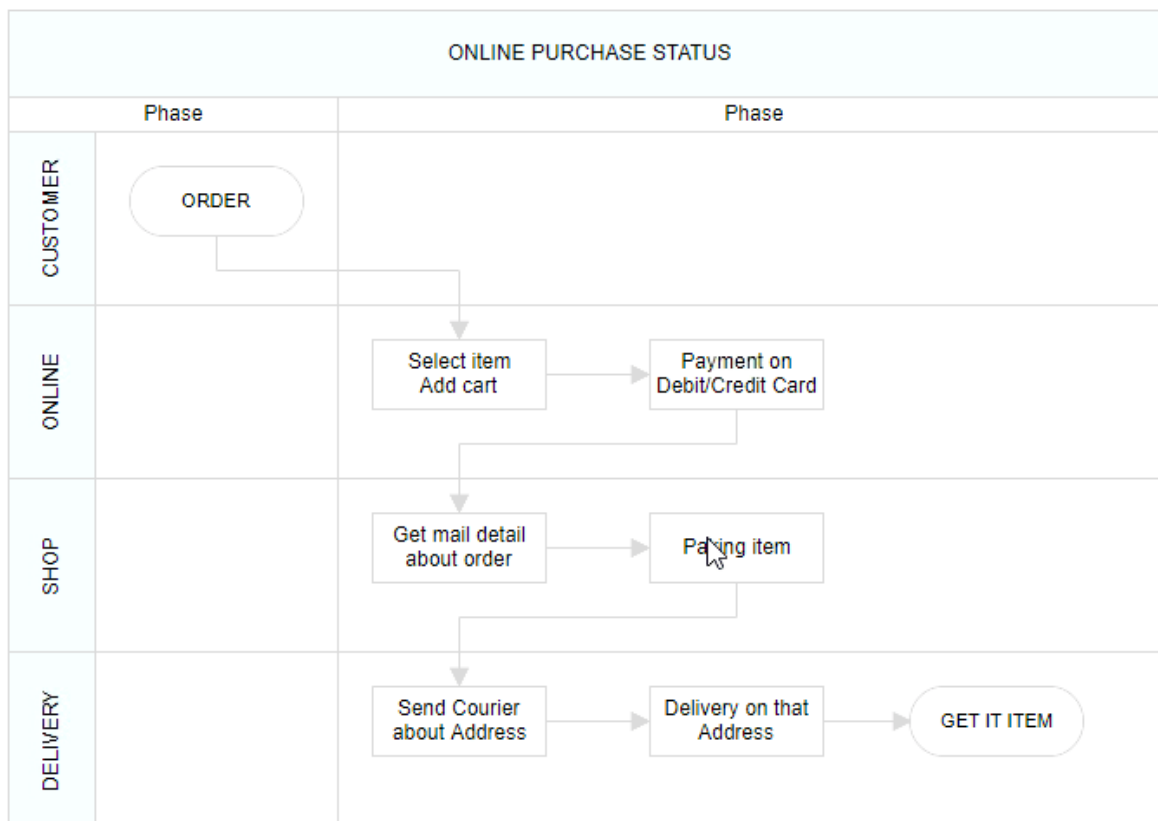
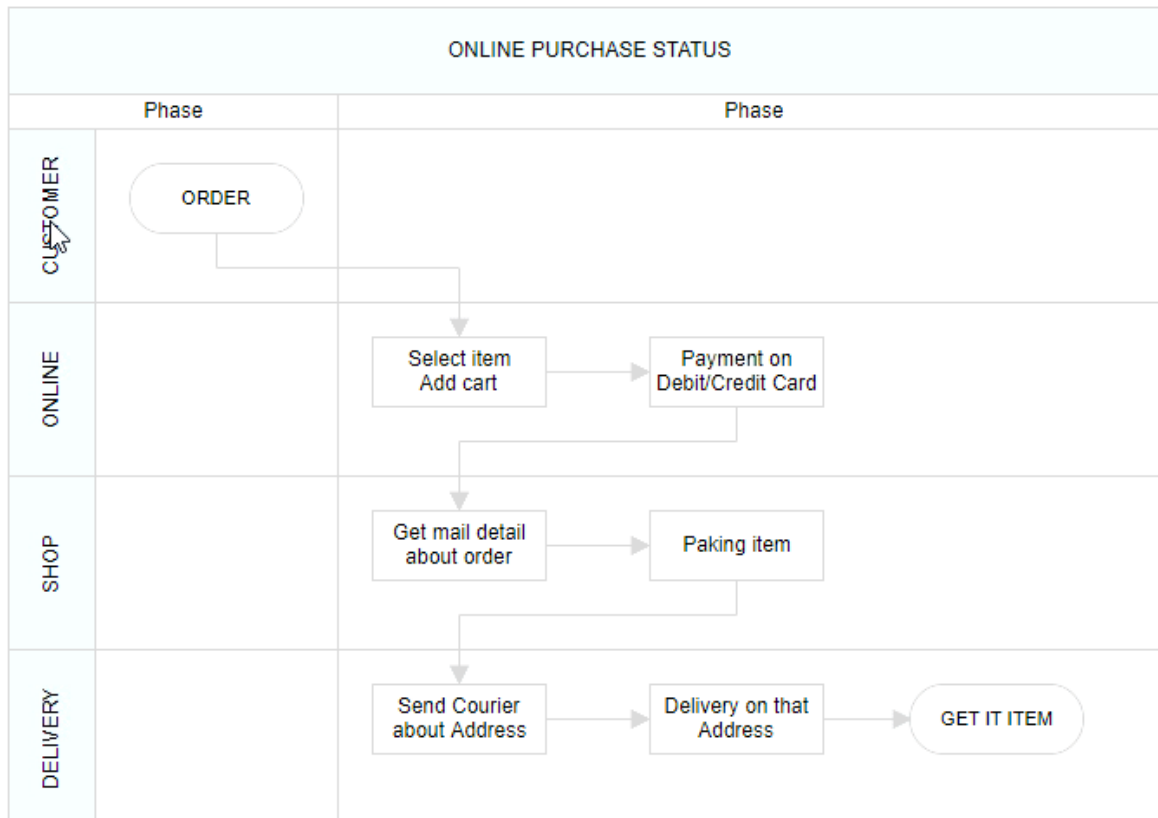
*Lane Header editing*

Diagram provides the support to edit Lane headers at runtime. The header editing can be achieved by double click event. Double clicking the header label will enable its editing.



Phase

Phase are the subprocess which will split each lane as horizontally or vertically based on the swimlane orientation. The multiple number of [Phase](#) can be added to swimlane.

PHASE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
        };
        List<Lane> Lanes = new List<Lane>();
        Lanes.Add(new Lane ()
        {
```

```

        Id = "stackCanvas1",
        Height = 100,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"Consumer" },
            Width = 50, Style = new DiagramTextStyle() { FontSize =
11 ,fill ="red"}
        },
    });
    List<Phase> Phases = new List<Phase>();
    Phases.Add(new Phase()
    {
        Id= "phase1",
        Offset=170,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"Phase" },
        },
    });
    swimlane.Shape = new SwimLane() {
        Type = "SwimLane",
        PhaseSize = 20,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
            Height = 50,
            Orientation = "Horizontal",
            Style = new DiagramTextStyle() { FontSize = 11 ,fill
="red"}
        },
        Lanes = Lanes,
        Phases = Phases
    };
    nodes.Add(swimlane);
    ViewBag.nodes = nodes;
    return View();
}
}

public class Node: DiagramNode {
    public string text;
}
}

```

Dynamically add phase to lane

You can add the phase at runtime by using client side API method called `addPhases`.

`typescript

```

let phase = [{
id: 'phase1', offset: 170,
header: { content: { content: 'Phase' } }

```



```
}}
```

```
diagram.addPhase(swimlane,phase)
```

```
,
```

Customizing phase

- The length of region can be set by using the [offset](#) property of the phase.
- Every phase region can be textually described with the [header](#) property of the phase.
- You can increase the width of phase by using [phaseSize](#) property of swimlane.
- The additional information can be provided to the phase by using the [addInfo](#) property of the phase.

PHASE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List<Lane> Lanes = new List<Lane>();
            Lanes.Add(new Lane()
            {
                Id = "stackCanvas1",
                Height = 100,
                Header = new Header()
                {
                    Annotation = new DiagramNodeAnnotation() { Content =
"Consumer" },
                    Width = 50,Style = new DiagramTextStyle() { FontSize =
11 ,fill ="red"}
                },
            });
            List<Phase> Phases = new List<Phase>();
            Phases.Add(new Phase()
            {
                Id= "phase1",
                Offset=170,
                Header = new Header()
                {
```

```

        Annotation = new DiagramNodeAnnotation() { Content =
"Phase" },
    },
    });
    swimlane.Shape = new SwimLane() {
        Type = "SwimLane",
        PhaseSize = 20,
        Header = new Header()
        {
            Annotation = new DiagramNodeAnnotation() { Content =
"ONLINE PURCHASE STATUS" },
            Height = 50,
            Orientation = "Horizontal",
            Style = new DiagramTextStyle() { FontSize = 11 ,fill
="red"}
        },
        Lanes = Lanes,
        Phases = Phases
    };
    nodes.Add(swimlane);
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

Phase interaction

Resizing

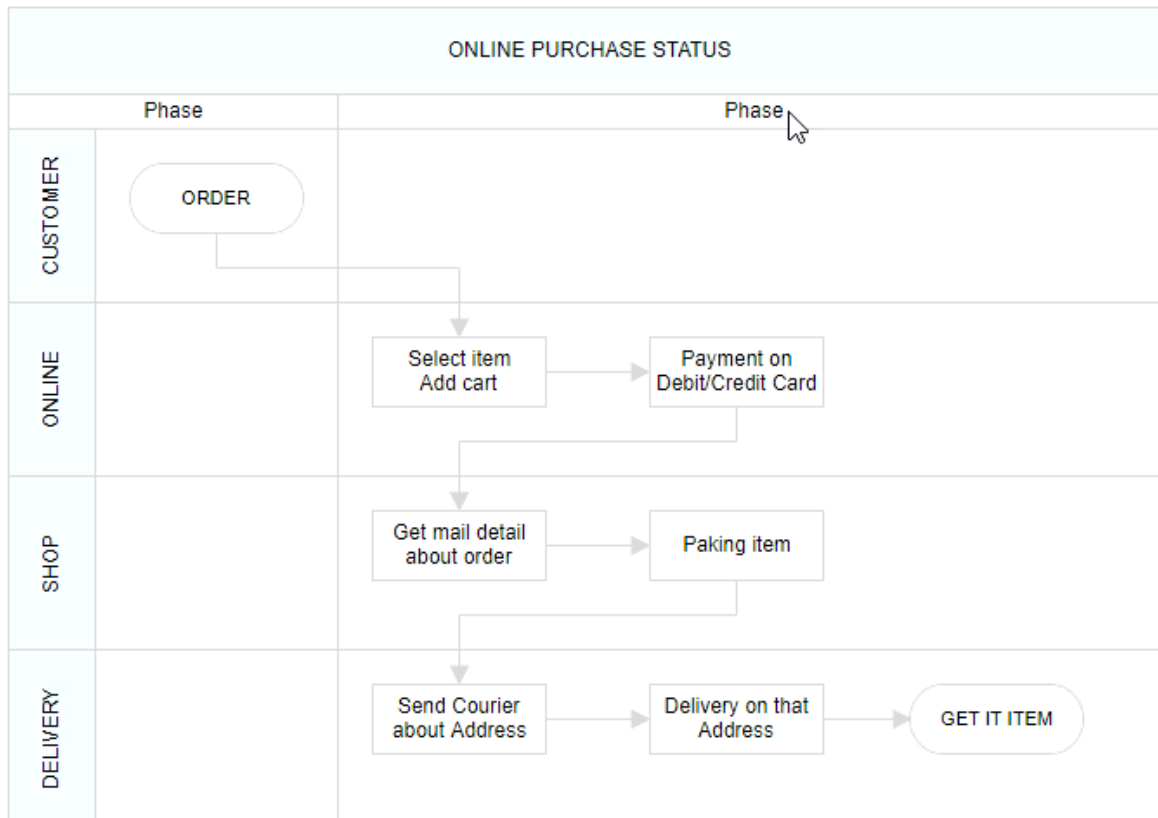
- The phase can be resized by using its selector.
- You must select the phase header to enable the phase selection.
- Once the phase can be resized, the lane size will be updated automatically.

Resizing helper

- The special resize selector will be used to resize the phase.
- The resize cursor will be available on the left and bottom direction for horizontal, and the top and bottom direction for vertical swimlane.

Phase header editing

Diagram provides the support to edit phase headers at runtime. The header editing is achieved by double click event. Double clicking the header label will enable the editing of that.



Add swimlane to palette

Diagram provides support to add swimlane and phases to symbol palette.

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
```

```

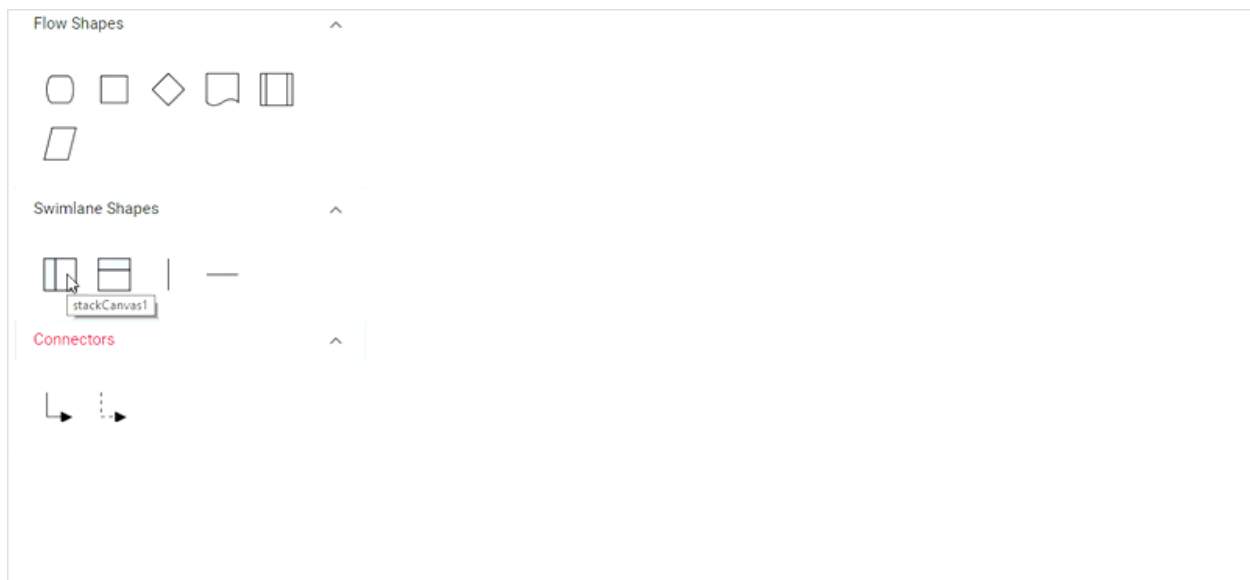
        BorderWidth=2,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1
    });
    ViewBag.nodes = nodes;
    return View();
}

public class Node: DiagramNode {
    public string text;
}
}

```

Drag and drop swimlane to palette

- The drag and drop support for swimlane shapes has been provided.
- When you drag and drop the lane shape, if the diagram already contains swimlane with the same orientation, the lane will be added and stacked inside a swimlane based on the order. Otherwise, it will be added a new swimlane.
- The phase will only drop on swimlane shape with same orientation.



Limitations

- Connectors cannot be canceled when added directly to swimlane. You must initialize the connector through connector collection.
- The phase line style cannot be edited.

Annotation in Diagram

[Annotation](#) is a block of text that can be displayed over a node or connector. Annotation is used to textually represent an object with a string that can be edited at runtime. Multiple annotations can be added to a node/connector.

<!-- markdownlint-disable MD033 -->

Create annotation

An annotation can be added to a node/connector by defining the annotation object and adding that to the annotation collection of the node/connector. The [content](#) property of annotation defines the text to be displayed.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "Node1", Style
= new DiagramTextStyle() { Color = "black", Fill = "transparent" } });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width =100,
                // add the Annotation for the Node
                Annotations = Node1,
```

```

    });
    // Sets the Annotation for the Connector
    List<DiagramConnectorAnnotation> Connector1 = new
List<DiagramConnectorAnnotation>();
    Connector1.Add(new DiagramConnectorAnnotation() { Content =
"Connector1", Style = new DiagramTextStyle() { Color = "black", Fill =
"transparent" } });
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() {
        Id = "connector1",
        SourceID = "Node1",
        TargetID = "Node2",
        // add the Annotation for the Connector
        Annotations = Connector1 });
    ViewBag.nodes = Nodes;
    ViewBag.connectors=Connectors;
    return View();
}
}
}

```

Add annotations at runtime

- Annotations can be added at runtime by using the client-side method [addLabels](#).
- The annotation's [ID](#) property is used to define the name of the annotation and its further used to find the annotation at runtime and do any customization.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
    }
}

```

```

public ActionResult Node()
{
    // Sets the Annotation for the Node
    List<DiagramNode> Nodes = new List<DiagramNode>();
    List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
    Node1.Add(new DiagramNodeAnnotation() { Content = "Node1", Style
= new DiagramTextStyle() { Color = "black", Fill = "transparent" } });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width =100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    ViewBag.nodes = Nodes;
    return View();
}
}
}

```

```
`javascript
```

```
var diagramElement = document.getElementById('element');
```

```
var diagram = diagramElement.ej2_instances[0];
```

```
var annotation = [{
```

```
id: 'label1',
```

```
//set the context
```

```
content: 'Annotation'
```

```
}]
```

```
//Method to add labels at run time
```

```
diagram.addLabels(diagram.nodes[0], annotation);
```

```
diagram.dataBind();
```

```
,
```

Remove annotation

A collection of annotations can be removed from the node by using client-side method [removeLabels](#).

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    
```

```
</div>
```

```
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "Node1", Style
= new DiagramTextStyle() { Color = "black", Fill = "transparent" } });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width =100,
                // add the Annotation for the Node
                Annotations = Node1,
            });
            ViewBag.nodes = Nodes;
            return View();
        }
    }
}
```

```
`javascript
```

```
var diagramElement = document.getElementById('element');
```

```
var diagram = diagramElement.ej2_instances[0];
```

```
var annotation = [{
```

```
id: 'label1',
```

```
content: 'Annotation'
```

```
}]
```

```
//Method to remove labels at runtime
```



```
diagram.removeLabels(diagram.nodes[0], annotation);
```

```
,
```

Update annotation at runtime

You can change any annotation properties at runtime and update it through the client-side method [dataBind](#).

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() { Content = "Node1", Style
= new DiagramTextStyle() { Color = "black", Fill = "transparent" } });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width =100,
                // add the Annotation for the Node
                Annotations = Node1,
            });
            ViewBag.nodes = Nodes;
            return View();
        }
    }
}
```

```
`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
diagram.nodes[0].annotations[0].content = 'Updated Annotation';
//Method to update the annotation at run time
diagram.dataBind();
`
```

Alignment

Annotation can be aligned relative to the node boundaries. It has [margin](#), [offset](#), horizontal, and vertical alignment settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset

The offset property of annotation is used to align the annotations based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Set the size for nodes annotation by using [width](#) and [height](#) properties.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
        }
    }
}
```

```

        List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
        Node1.Add(new DiagramNodeAnnotation() {
            //Sets the offset for the content
            Content = "Node1",
            Style = new DiagramTextStyle() {
                Color = "black",
                Fill = "transparent" },
            //Sets the offset for the content
            Offset = new DiagramPoint() {X=0,Y=1}
        });
        Nodes.Add(new DefaultNode()
        {
            Id = "Node1",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width =100,
            // add the Annotation for the Node
            Annotations = Node1,
        });
        ViewBag.nodes = Nodes;
        return View();
    }
}

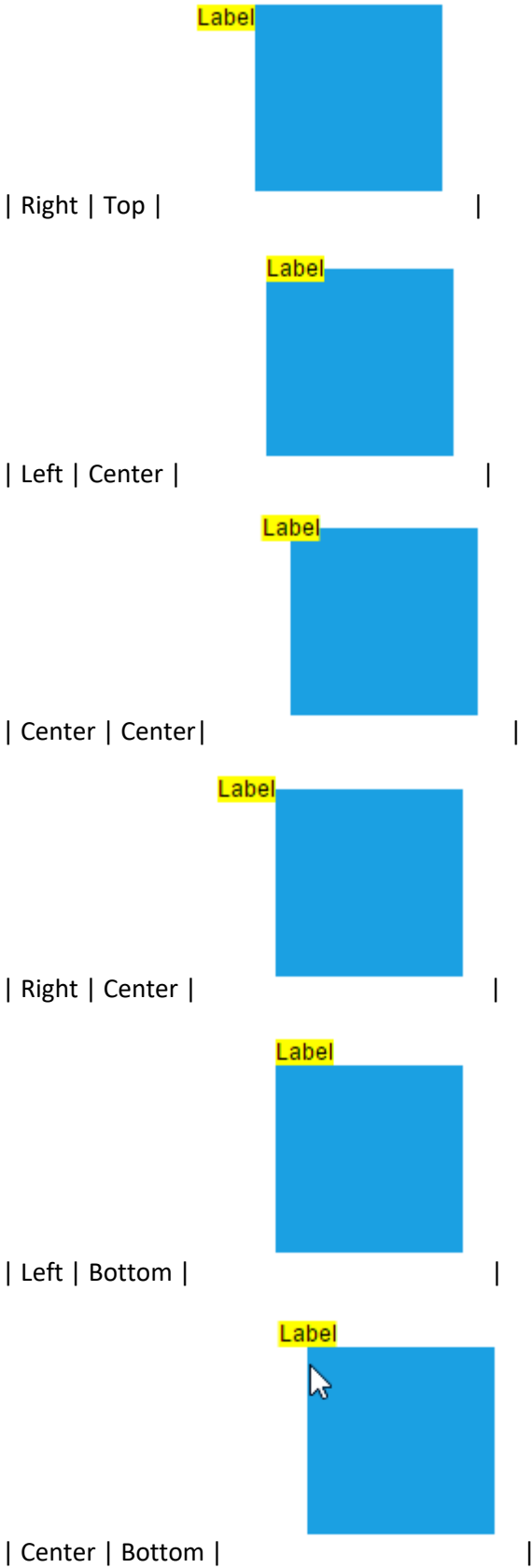
```

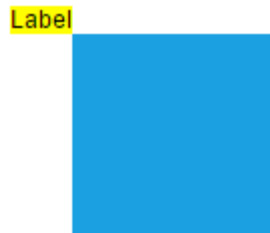
Horizontal and vertical alignment

The [horizontalAlignment](#) property of annotation is used to set how the annotation is horizontally aligned at the annotation position determined from the fraction values. The [verticalAlignment](#) property is used to set how annotation is vertically aligned at the annotation position.

The following tables illustrates all the possible alignments visually with 'offset (0, 0)'.

Horizontal Alignment	Vertical Alignment	Output with Offset(0,0)
Left	Top	
Center	Top	





| Right | Bottom |

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() {
                //Sets the offset for the content
                Content = "Node1",
                Style = new DiagramTextStyle() {
                    Color = "black",
                    Fill = "transparent" },
                // Sets the horizontal alignment as left
                HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
                // Sets the vertical alignment as Center
                VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center
            });
            Nodes.Add(new DefaultNode()
            {
```

```

        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    ViewBag.nodes = Nodes;
    return View();
}
}
}

```

Annotation alignment with respect to segments

The offset and alignment properties of annotation allows to align the connector annotations with respect to the segments.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
            List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() {
                //Sets the offset for the content
                Content = "Node1",
                Style = new DiagramTextStyle() {
                    Color = "black",

```

```

        Fill = "transparent" },
    });
    List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
    Node2.Add(new DiagramNodeAnnotation()
    {
        //Sets the offset for the content
        Content = "Node1",
        Style = new DiagramTextStyle()
        {
            Color = "black",
            Fill = "transparent"
        },
    });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node2",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node2,
    }); // Sets the Annotation for the Connector
    List<DiagramConnectorAnnotation> Connector1 = new
List<DiagramConnectorAnnotation>();
    Connector1.Add(new DiagramConnectorAnnotation() {
        Content = "Connector1",
        // Sets the offset for the content
        Offset = 0,
        Style = new DiagramTextStyle() { Color = "white", Fill =
"transparent" } });
    Connector1.Add(new DiagramConnectorAnnotation()
    {
        Content = "Connector2",
        // Sets the offset for the content
        Offset = 1,
        Style = new DiagramTextStyle() { Color = "white", Fill =
"transparent" }
    }); List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector()
    {
        Id = "connector1",
        SourceID = "Node1",
        TargetID = "Node2",
        // add the Annotation for the Connector

```

```

        Annotations = Connector1
    });
    ViewBag.nodes = Nodes;
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

Margin

[Margin](#) is an absolute value used to add some blank space in any one of its four sides. The annotations can be displaced with the margin property. The following code example illustrates how to align a annotation based on its `offset`, `horizontalAlignment`, `verticalAlignment`, and `margin` values.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
            .Width("100%")
            .Height("500px")
            .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
            .Nodes(ViewBag.nodes)
            .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() {
                //Sets the offset for the content
                Content = "Node1",
                Style = new DiagramTextStyle() {
                    Color = "black",
                    Fill = "transparent" },
                //Sets the margin for the annotation
                Margin = new DiagramMargin() { Top = 10 }
            });
        }
    }
}

```



```

        // Sets the horizontal alignment as left
        HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
        // Sets the vertical alignment as Center
        VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center
    });
    List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
    Node2.Add(new DiagramNodeAnnotation()
    {
        //Sets the offset for the content
        Content = "Node1",
        Style = new DiagramTextStyle()
        {
            Color = "black",
            Fill = "transparent"
        },
        //Sets the margin for the annotation
        Margin = new DiagramMargin() { Bottom = 10 },
        // Sets the horizontal alignment as left
        HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
        // Sets the vertical alignment as Center
        VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node2",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node2,
    });
    // Sets the Annotation for the Connector
    List<DiagramConnectorAnnotation> Connector1 = new
List<DiagramConnectorAnnotation>();
    Connector1.Add(new DiagramConnectorAnnotation() {
        Content = "Connector1",
        // Sets the offset for the content
        Offset = 0,
        //Sets the margin for the annotation
        Margin = new DiagramMargin() { Top = 10 },
        Style = new DiagramTextStyle() { Color = "white", Fill =
"transparent" } });

```

```

        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector()
        {
            Id = "connector1",
            SourceID = "Node1",
            TargetID = "Node2",
            // add the Annotation for the Connector
            Annotations = Connector1
        });
        ViewBag.nodes = Nodes;
        ViewBag.connectors = Connectors;
        return View();
    }
}

```

Text align

The [textAlign](#) property of annotation allows to set how the text should be aligned (left, right, center, or justify) inside the text block.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation()
            {

```

```

        //Sets the offset for the content
        Content = "Text align is set as Left",
        Style = new DiagramTextStyle()
        {
            Color = "black",
            Fill = "transparent",
            // Sets the textAlign as left for the content
            TextAlign = Syncfusion.EJ2.Diagrams.TextAlign.Left
        },
    });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    ViewBag.nodes = Nodes;
    return View();
}
}

```

Hyperlink

Diagram provides a support to add a [hyperlink](#) for the nodes/connectors annotation. It can also be customized.

A user can open the hyperlink in the new window, the same tab and the new tab by using the [hyperlinkOpenState](#) property.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

HYPERLINK.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

```

```

using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() {
                //Sets the offset for the content
                Content = "Node1",
                Style = new DiagramTextStyle() {
                    Color = "black",
                    Fill = "transparent" },
                //Sets the margin for the annotation
                Margin = new DiagramMargin() { Top = 10 }
                // Sets the horizontal alignment as left
                HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
                // Sets the vertical alignment as Center
                VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center
            });
            List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
            Node2.Add(new DiagramNodeAnnotation()
            {
                //Sets the offset for the content
                Content = "Node1",
                hyperlink= {link= "https://www.google.com/"},
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Fill = "transparent"
                },
                //Sets the margin for the annotation
                Margin = new DiagramMargin() { Bottom = 10 }
                // Sets the horizontal alignment as left
                HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
                // Sets the vertical alignment as Center
                VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width =100,
                // add the Annotation for the Node
                Annotations = Node1,
            });
        }
    }
}

```

```

Nodes.Add(new DefaultNode()
{
    Id = "Node2",
    OffsetY = 100,
    OffsetX = 100,
    Height = 100,
    Width = 100,
    // add the Annotation for the Node
    Annotations = Node2,
}); // Sets the Annotation for the Connector
List<DiagramConnectorAnnotation> Connector1 = new
List<DiagramConnectorAnnotation>();
Connector1.Add(new DiagramConnectorAnnotation() {
    Content = "Connector1",
    // Sets the offset for the content
    Offset = 0,
    //Sets the margin for the annotation
    Margin = new DiagramMargin() { Top = 10 },
    Style = new DiagramTextStyle() { Color = "white", Fill =
"transparent" } });
List<DiagramConnector> Connectors = new
List<DiagramConnector>();
Connectors.Add(new DiagramConnector()
{
    Id = "connector1",
    SourceID = "Node1",
    TargetID = "Node2",
    // add the Annotation for the Connector
    Annotations = Connector1
});
ViewBag.nodes = Nodes;
ViewBag.connectors = Connectors;
return View();
}
}
}

```

Template Support for Annotation

Diagram provides template support for annotation. You should define a SVG/HTML content as string in the annotation's [template](#) property.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

LABELTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation() {
                //Sets the offset for the content
                Content = "Node1",
                Style = new DiagramTextStyle() {
                    Color = "black",
                    Fill = "transparent"
                },
                //Set an template for Node annotation
                Template = "<div><input type='button'
value='Submit'></div>"
            });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width = 100,
                // add the Annotation for the Node
                Annotations = Node1,
            });
            // Sets the Annotation for the Connector
            List<DiagramConnectorAnnotation> Connector1 = new
List<DiagramConnectorAnnotation>();
            Connector1.Add(new DiagramConnectorAnnotation() {
                Content = "Connector1",
                // Sets the offset for the content
                Offset = 0,
                //Set an template for Connector annotation
                Template = "<div><input type='button' value='Submit'></div>"
            });
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector()
            {
                Id = "connector1",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
                TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
            }

```

```

        // add the Annotation for the Connector
        Annotations = Connector1
    });
    ViewBag.nodes = Nodes;
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

Wrapping

When text overflows node boundaries, you can control it by using [text wrapping](#). So, it is wrapped into multiple lines. The wrapping property of annotation defines how the text should be wrapped.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation()
            {
                Content= "Annotation Text Wrapping",
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Fill = "transparent",
                    // Sets the TextWrapping as wrap for the content
                    TextWrapping = Syncfusion.EJ2.Diagrams.TextWrap.Wrap

```

```

    },

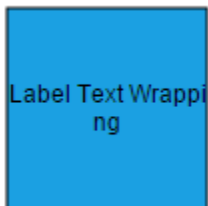
    });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    ViewBag.nodes = Nodes;
    return View();
}
}

```



| No Wrap | Text will not be wrapped. |

| Wrap | Text-wrapping occurs, when the text overflows beyond the available node width. |



| WrapWithOverflow (Default) | Text-wrapping occurs, when the text overflows beyond the available node width. However, the text may overflow beyond the node width in the case of a very long word. |



Text overflow

The label's [TextOverflow](#) property is used control whether to display the overflowed content in node or not.

CSHTML


```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Model = new
List<DiagramNodeAnnotation>();
            Model.Add(new DiagramNodeAnnotation()
            {
                Content= "Annotation Text Wrapping",
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Fill = "transparent",
                    // Sets the TextOverflow as Ellipsis for the content
                    TextOverflow =
Syncfusion.EJ2.Diagrams.TextOverflow.Ellipsis
                },
            });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width = 100,
                // add the Annotation for the Node
                Annotations = Model,
            });
            ViewBag.nodes = Nodes;
            return View();
        }
    }
}

```

```
}
}
```

Appearance

- You can change the font style of the annotations with the font specific properties (fontSize, fontFamily, color).
- The label's [bold](#), [italic](#), and [textDecoration](#) properties are used to style the label's text.
- The label's [fill](#), [strokeColor](#), and [strokeWidth](#) properties are used to define the background color and border color of the annotation and the [opacity](#) property is used to define the transparency of the annotations.
- The [visible](#) property of the annotation enables or disables the visibility of annotation.

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Model = new
            List<DiagramNodeAnnotation>();
            Model.Add(new DiagramNodeAnnotation()
            {
                Content = "Annotation Text Wrapping",
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Bold = true,
                    Italic = true,
                    FontSize = 12,
                }
            })
        }
    }
}
```

```

        FontFamily="TimesNewRoman",
        Fill = "transparent",
    },
    });
    Nodes.Add(new DefaultNode()
    {
        Id = "Node1",
        OffsetY = 100,
        OffsetX = 100,
        Height = 100,
        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    ViewBag.nodes = Nodes;
    return View();
}
}

```

The fill, border, and opacity appearances of the text can also be customized with appearance specific properties of annotation.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
            List<DiagramNodeAnnotation>();

```

```

        Node1.Add(new DiagramNodeAnnotation()
        {
            Content = "Annotation Text Wrapping",
            Constraints =
Syncfusion.EJ2.Diagrams.AnnotationConstraints.ReadOnly,
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent",
            },

        });
        Nodes.Add(new DefaultNode()
        {
            Id = "Node1",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width = 100,
            // add the Annotation for the Node
            Annotations = Node1,
        });
        ViewBag.nodes = Nodes;
        return View();
    }
}

```

Interaction

Diagram allows annotation to be interacted by selecting, dragging, rotating, and resizing. Annotation interaction is disabled, by default. You can enable annotation interaction with the [constraints](#) property of annotation. You can also curtail the services of interaction by enabling either selecting, dragging, rotating, or resizing individually with the respective constraints property of annotation.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;

```

```

namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Model = new
List<DiagramNodeAnnotation>();
            Model.Add(new DiagramNodeAnnotation()
            {
                Content = "Annotation Text Wrapping",
                Constraints =
Syncfusion.EJ2.Diagrams.AnnotationConstraints.ReadOnly,
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Fill = "transparent",
                },
            });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width = 100,
                // add the Annotation for the Node
                Annotations = Model,
            });
            ViewBag.nodes = Nodes;
            return View();
        }
    }
}

```

Edit

Diagram provides support to edit an annotation at runtime, either programmatically or interactively. By default, annotation is in view mode. But it can be brought to edit mode in two ways;

- Programmatically - By using [startTextEdit](#) method, edit the text through programmatically.
- Interactively
 1. By double-clicking the annotation.
 2. By selecting the item and pressing the F2 key.

Double-clicking any annotation will enable editing and the node enables first annotation editing. When the focus of editor is lost, the annotation for the node is updated. When you double-click on the node/connector/diagram model, the [doubleClick](#) event gets triggered.

Read-only annotations

Diagram allows to create read-only annotations. You have to set the read-only property of annotation to enable/disable the read-only [constraints](#).

CSHTML

```
<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();
            List<DiagramNodeAnnotation> Node1 = new
            List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation()
            {
                Content = "Annotation Text Wrapping",
                Style = new DiagramTextStyle()
                {
                    Color = "black",
                    Bold = true,
                    Italic = true,
                    FontSize = 12,
                    FontFamily="TimesNewRoman",
                    Fill = "transparent",
                },
            });
            Nodes.Add(new DefaultNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
```

```

        Width = 100,
        // add the Annotation for the Node
        Annotations = Node1,
    });
    ViewBag.nodes = Nodes;
    return View();
}
}
}

```

Drag Limit

- The diagram control now supports defining the [dragLimit](#) to the label while dragging from the connector and also update the position to the nearest segment offset.
- You can set the value to dragLimit [left](#), [right](#), [top](#), and [bottom](#) properties which allows the dragging of connector labels to a certain limit based on the user defined values.
- By default, drag limit will be disabled for the connector. It can be enabled by setting connector constraints as drag.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Connectors(ViewBag.connectors).Render())
    </div>
</div>

```

DRAGLIMIT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Connector
            List<DiagramConnectorAnnotation> Connector1 = new
            List<DiagramConnectorAnnotation>();
            Connector1.Add(new DiagramConnectorAnnotation() {
                Content = "Connector1",
                // Sets the offset for the content
                Offset = 0,
            });
        }
    }
}

```

```

        //Enables drag constraints for a connector.
        Constraints = AnnotationConstraints.Interaction |
AnnotationConstraints.Drag,
        //Set drag limit for a connector annotation.
        dragLimit = new
DiagramMargin() {Left=20,Right=20,Top=20,Bottom=20}
    });
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector()
    {
        Id = "connector1",
        SourcePoint = new DiagramPoint() { X = 100, Y = 100 },
        TargetPoint = new DiagramPoint() { X = 200, Y = 200 },
        // add the Annotation for the Connector
        Annotations = Connector1
    });
    ViewBag.connectors = Connectors;
    return View();
}
}
}

```

Multiple annotations

You can add any number of annotations to a node or connector.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("container")
        .Width("100%")
        .Height("500px")
        .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
        .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();

```



```

        List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
        Node1.Add(new DiagramNodeAnnotation()
        {
            Content = "Annotation Text Wrapping",
            Offset = new DiagramPoint() { X=0,Y=0},
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent",
            },
        });
        Node1.Add(new DiagramNodeAnnotation()
        {
            Content = "Annotation Text Wrapping",
            Offset = new DiagramPoint() { X = 0.5, Y = 0.5 },
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent",
            },
        });
        Node1.Add(new DiagramNodeAnnotation()
        {
            Content = "Annotation Text Wrapping",
            Offset = new DiagramPoint() { X = 1, Y = 1 },
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent",
            },
        });
        Nodes.Add(new DefaultNode()
        {
            Id = "Node1",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width = 100,
            // add the Annotation for the Node
            Annotations = Node1,
        });
        ViewBag.nodes = Nodes;
        return View();
    }
}

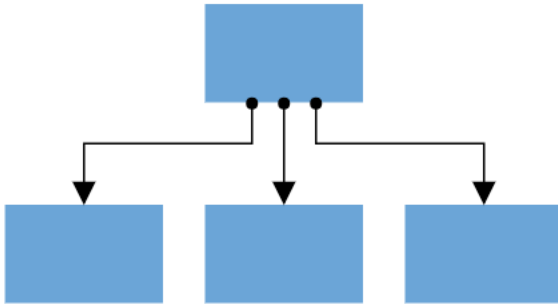
```

Constraints

The constraints property of annotation allows to enable or disable certain annotation behaviours. For instance, you can disable annotation editing.

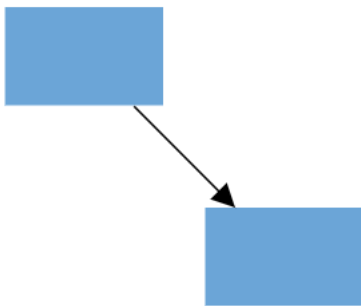
Ports in Diagram Control

Diagram provides support to define custom ports for making connections.

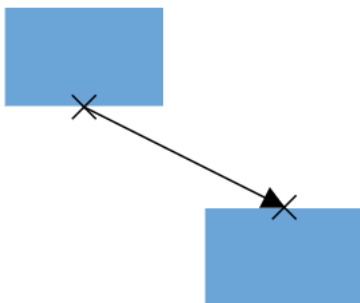


<!-- markdownlint-disable MD033 -->

When a connector is connected between two nodes, its end points are automatically docked to the node's nearest boundary.



Ports act as the connection points of the node and allows to create connections with only those specific points.



Create port

Add ports when initializing nodes

To add a connection port, define the port object and add it to node's ports collection. The `offset` property of port accepts an object of fractions and used to determine the position of ports.

PORT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramPort > ports1 = new List < DiagramPort > ();
            ports1.Add(new CustomPort() {
                Id = "port1", Shape = PortShapes.Circle,
                Offset = new DiagramPoint() {
                    X = 0.5, Y = 0.5
                }
            });
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Ports = ports1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Add ports at runtime

Add ports at runtime by using the client-side method [addPorts](#).

The port's ID property is used to define the unique ID for the port and its further used to find the port at runtime. If ID is not set, then default ID is automatically set.

RUN.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;

```

```

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```
// Initialize port collection
```

```
var port= [{
```

```
id: 'port1',
```

```
offset: {
```

```
x: 0,
```

```
y: 0.5
```

```
},
```

```
visibility: PortVisibility.Visible
```

```
}{
```

```
id: 'port2',
```

```
offset: {
```

```
x: 1,
```

```
y: 0.5
```

```
},
```

```
visibility: PortVisibility.Visible
```

```
},
```

```
{
```

```

id: 'port3',
offset: {
x: 0.5,
y: 0
},
visibility: PortVisibility.Visible
},
{
id: 'port4',
offset: {
x: 0.5,
y: 1
},
visibility: PortVisibility.Visible
}
];
// Method to add ports through run time
diagram.addPorts(diagram.nodes[0], port);
,

```

Remove ports at runtime

Remove ports at runtime by using client-side method [removePorts](#).

REMOVE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramPort > ports1 = new List < DiagramPort > ();
            ports1.Add(new CustomPort() {
                Id = "port1", Shape = PortShapes.Circle, Offset = new
DiagramPoint() {
                    X = 0, Y = 0.5
                }, Visibility = PortVisibility.Visible
            });
            ports1.Add(new CustomPort() {

```

```

        Id = "port2", Shape = PortShapes.Circle, Offset = new
DiagramPoint() {
            X = 1, Y = 0.5
        }, Visibility = PortVisibility.Visible
    });
    ports1.Add(new CustomPort() {
        Id = "port3", Shape = PortShapes.Circle, Offset = new
DiagramPoint() {
            X = 0.5, Y = 1
        }, Visibility = PortVisibility.Visible
    });
    ports1.Add(new CustomPort() {
        Id = "port4", Shape = PortShapes.Circle, Offset = new
DiagramPoint() {
            X = 1, Y = 1
        }, Visibility = PortVisibility.Visible
    });
    List < DiagramNode > nodes = new List < DiagramNode > ();
    nodes.Add(new Node() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "#6BA5D7",
            StrokeColor = "White"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
        Ports = ports1
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```

var ports = [{
  id: 'port1',
}, {
  id: 'port2',
}, {
  id: 'port3',
}, {
  id: 'port4',

```

```

}}
diagram.removePorts(diagram.nodes[0], ports);
,

```

Update port at runtime

You can change any port properties at runtime and update it through the client-side method [dataBind](#).

PORT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramPort > ports1 = new List < DiagramPort > ();
            ports1.Add(new CustomPort() {
                Id = "port1", Shape = PortShapes.Circle,
                Offset = new DiagramPoint() {
                    X = 0.5, Y = 0.5
                }
            });
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Ports = ports1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```

diagram.nodes[0].ports[0].offset = {
x: 1,

```

y: 1

};

diagram.dataBind();

,

Appearance

- The shape of port can be changed by using its shape property. To explore the different types of port shapes, refer to Port Shapes. If you need to render a custom shape, then you can set shape as path and define path using path data property of port.
- The appearance of ports can be customized by using [strokeColor](#), [strokeWidth](#), and [fill](#) properties of the port.
- Customize the port size by using the [width](#) and [height](#) properties of port.
- The ports [visibility](#) property allows to define, when the port should be visible.

APPEAR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramPort > ports1 = new List < DiagramPort > ();
            ports1.Add(new CustomPort() {
                Id = "port1", Shape = PortShapes.Circle,
                Offset = new DiagramPoint() {
                    X = 0.5, Y = 0.5
                },
                style = new DiagramShapeStyle() {
                    Fill = "Black", StrokeColor = "Red", StrokeWidth = 2
                },
                Visibility = PortVisibility.Visible
            });
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Ports = ports1
            });
        }
    }
}
```



```

        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

Offset

The offset property of port is used to align the port based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Constraints

The constraints property allows to enable/disable certain behaviors of ports. For more information about port constraints, refer to [Port Constraints](#).

Constraints

Constraints are used to enable or disable certain behaviors of the diagram, nodes and connectors. Constraints are provided as flagged enumerations, so that multiple behaviors can be enabled or disabled using Bitwise operators (&, |, ~, <<, etc.).

To know more about Bitwise operators, refer to [Bitwise Operations](#).

Diagram constraints

Diagram constraints allows to enable or disable the following behaviors:

- Page editing
- Bridging
- Zoom and pan
- Undo/redo
- Tooltip

DEFAULT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                }
            });
        }
    }
}

```

```

        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

For more information about diagram constraints, refer to [DiagramConstraints](#).

Node constraints

Node constraints allows to enable or disable the following behaviors of node. They are as follows:

- Selection
- Deletion
- Drag
- Resize
- Rotate
- Connect
- Shadow
- Tooltip

NODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            // Set constraints for the node

```

```

        constraints = NodeConstraints.Default &
~NodeConstraints.Rotate
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

For more information about node constraints, refer to [NodeConstraints](#).

Connector constraints

Connector constraints allows to enable or disable certain behaviors of connectors.

- Selection
- Deletion
- Drag
- Segment editing
- Tooltip
- Bridging

CONNECTOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramConnector > connectors = new List <
DiagramConnector > ();
            connectors.Add(new DiagramConnector() {
                Id = "connector1",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100},
                TargetPoint = new DiagramPoint() { X = 300, Y = 200},
                Type = Segments.Orthogonal,
                // Set constraints for the connector
                constraints = ConnectorConstraints.Default &
~ConnectorConstraints.Select
            });
            ViewBag.Connectors = connectors;
            return View();
        }
    }
}

```

For more information about connector constraints, refer to [ConnectorConstraints](#).

Port constraints

You can enable or disable certain behaviors of port. They are as follows:

- Connect
- ConnectOnDrag

PORT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramPort > ports1 = new List < DiagramPort > ();
            ports1.Add(new CustomPort() {
                Id = "port1", Shape = PortShapes.Circle,
                Offset = new DiagramPoint() {
                    X = 0.5, Y = 0.5
                },
                Constraints = PortConstraints.Nones
            });
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Ports = ports1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

For more information about port constraints, refer to [PortConstraints](#).

Annotation constraints

You can enable or disable read-only mode for the annotations by using the annotation constraints.

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }, Constraints = AnnotationConstraints.ReadOnly
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

For more details about annotation constraints, refer to [AnnotationConstraints](#).

Selector constraints

Selector visually represents the selected elements with certain editable thumbs. The visibility of the thumbs can be controlled with selector constraints. The part of selector is categorized as follows:

- Resizer
- Rotator
- User handles

SELECTOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

For more information about selector constraints, refer to [SelectorConstraints](#).

Snap constraints

Snap constraints control the visibility of gridlines and enable or disable snapping. Snap constraints allow to set the following behaviors.

- Show only horizontal or vertical gridlines.
- Show both horizontal and vertical gridlines.
- Snap to either horizontal or vertical gridlines.
- Snap to both horizontal and vertical gridlines.

SNAP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
```

```

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

For more information about snap constraints, refer to [SnapConstraints](#).

Boundary constraints

Boundary constraints defines a boundary for the diagram inside which the interaction should be done. Boundary constraints allow to set the following behaviors.

- Infinite boundary
- Diagram sized boundary
- Page sized boundary

PAGE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {

```

```

        Fill = "#6BA5D7",
        StrokeColor = "White"
    },
    text = "node1",
    OffsetX = 100,
    OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

For more information about selector constraints, refer to [BoundaryConstraints](#).

Inherit behaviors

Some of the behaviors can be defined through both the specific object (node/connector) and diagram. When the behaviors are contradictorily defined through both, the actual behavior is set through inherit options.

BRIDGING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramConnector > connectors = new List <
DiagramConnector > ();
            connectors.Add(new DiagramConnector() {
                Id = "connector1",
                SourcePoint = new DiagramPoint() { X = 100, Y = 100},
                TargetPoint = new DiagramPoint() { X = 300, Y = 200},
                Type = Segments.Orthogonal,
                // Set constraints for the connector
                Constraints = ConnectorConstraints.Default &
~ConnectorConstraints.InheritBridging
            });
            ViewBag.Connectors = connectors;
            return View();
        }
    }
}

```


Bitwise operations

Bitwise operations are used to manipulate the flagged enumerations [enum]. In this section, Bitwise operations are illustrated by using node constraints. The same is applicable while working with node constraints, connector constraints, or port constraints.

Add operation

You can add or enable multiple values at a time by using Bitwise '|' (OR) operator.

`typescript

```
node.constraints = NodeConstraints.Select | NodeConstraints.Rotate;
```

,

In the previous example, you can do both the selection and rotation operation.

Remove Operation

You can remove or disable values by using Bitwise '&~' (XOR) operator.

`typescript

```
node.constraints = node.constraints & ~(NodeConstraints.Rotate);
```

,

In the previous example, rotation is disabled but other constraints are enabled.

Check operation

You can check any value by using Bitwise '&' (AND) operator.

`typescript

```
if ((node.constraints & (NodeConstraints.Rotate)) == (NodeConstraints.Rotate));
```

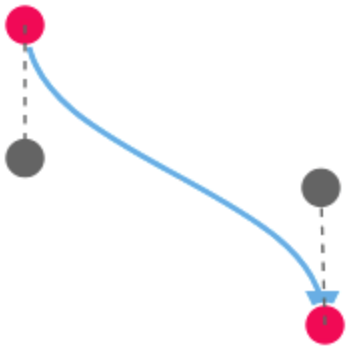
,

In the previous example, check whether the rotate constraints are enabled in a node. When node constraints have rotate constraints, the expression returns a rotate constraint.

Interaction

Bezier segment thumbs

- Bezier segments are annotated with two thumbs to represent the control points. Control points of the curve can be configured by clicking and dragging the control thumbs.

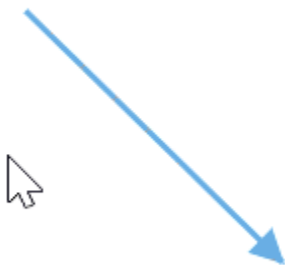


- Each segment of a selected connector is editable with some specific handles/thumbs.

For connector editing, you have to inject the `[ConnectorEditing]()` module.

End point handles in connector

Source and target points of the selected connectors are represented with two handles. Clicking and dragging those handles helps to adjust the source and target points.

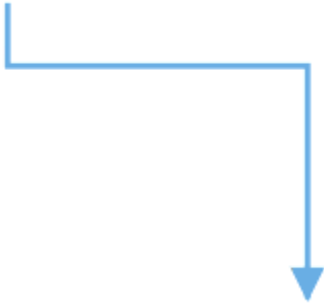


- If you drag the connector end points, then the following events can be used to do your customization.
- When the connector source point is changed, the [sourcePointChange](#) event gets triggered.
- When the connector target point is changed, the [targetPointChange](#) event gets triggered.
- When you connect connector with ports/node or disconnect from it, the [connectionChange](#) event gets triggered.

Orthogonal segment thumbs

- Orthogonal thumbs allows to adjust the length of adjacent segments by clicking and dragging it.

- When necessary, some segments are added or removed automatically, when dragging the segment. This is to maintain proper routing of orthogonality between segments.

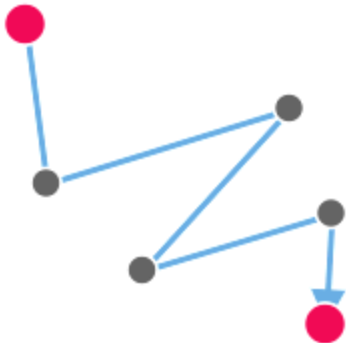


Straight segment editing

- End point of each straight segment is represented by a thumb that enables to edit the segment.
- Any number of new segments can be inserted into a straight line by clicking, when Shift and Ctrl keys are pressed (Ctrl+Shift+Click).



- Straight segments can be removed by clicking the segment end point, when Ctrl and Shift keys are pressed (Ctrl+Shift+Click).

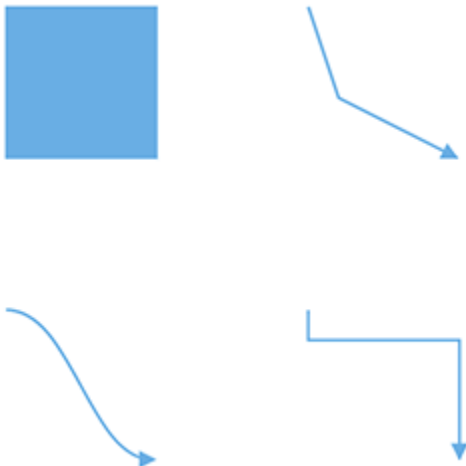


Selection in diagram

Selector provides a visual representation of selected elements. It behaves like a container and allows to update the size, position, and rotation angle of the selected elements through interaction and by using program. Single or multiple elements can be selected at a time.

Single selection

An element can be selected by clicking that element. During single click, all previously selected items are cleared.



- While selecting the diagram elements, the following events can be used to do your customization.
- When selecting or unselecting the diagram elements, the [selectionChange](#) event gets triggered.

Selecting a group

When a child element of any group is clicked, its contained group is selected instead of the child element. With consecutive clicks on the selected element, selection is changed from top to bottom in the hierarchy of parent group to its children.

Multiple selection

Multiple elements can be selected with the following ways:

- Ctrl+Click - During single click, any existing item in the selection list will be cleared, and only the item clicked recently will be there in the selection list. To avoid cleaning the old selected item, Ctrl key must be on hold when clicking.
- Selection rectangle/rubber band selection - Clicking and dragging the diagram area allows to create a rectangular region. The elements that are covered under the rectangular region are selected at the end.

![Multiple Rubberband Selection](../images/Multiselect_Highlight.JPG)

Select/Unselect elements using program

The client-side methods [select](#) and [clearSelection](#) helps to select or clear the selection of the elements at runtime.

Get the current selected items from the [nodes](#) and [connectors](#) collection of the [selectedItems](#) property of the diagram model.

Select entire elements in diagram programmatically

The client-side method [selectAll](#) is used to select all the elements such as nodes/connectors in the diagram. Refer to the following link which shows how to use [selectAll](#) method on the diagram.

Drag and drop nodes over other elements

Diagram provides support to drop a node/connector over another node/connector. The [drop](#) event is raised to notify that an element is dropped over another one and it is disabled, by default. It can be enabled with the constraints property.

User handles

- User handles are used to add some frequently used commands around the selector. To create user handles, define and add them to the [userHandles](#) collection of the [selectedItems](#) property.
- The name property of user handle is used to define the name of the user handle and its further used to find the user handle at runtime and do any customization.

Alignment

User handles can be aligned relative to the node boundaries. It has [margin](#), [offset](#), [side](#), [horizontalAlignment](#), and [verticalAlignment](#) settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset

The [offset](#) property of [userHandles](#) is used to align the user handle based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Side

The [side](#) property of [userHandles](#) is used to align the user handle by using the [Top](#), [Bottom](#), [Left](#), and [Right](#) options.

Horizontal and vertical alignments

The [horizontalAlignment](#) property of [userHandles](#) is used to set how the user handle is horizontally aligned at the position based on the [offset](#). The [verticalAlignment](#) property is used to set how user handle is vertically aligned at the position.

Margin

Margin is an absolute value used to add some blank space in any one of its four sides. The [userHandles](#) can be displaced with the [margin](#) property.

Notification for the mouse button clicked

The diagram component notifies the mouse button clicked. For example, whenever the right mouse button is clicked, the clicked button is notified as right. The mouse click is notified with,

Notification	Description
Left	When the left mouse button is clicked, left is notified
Middle	When the mouse wheel is clicked, middle is notified
Right	When the right mouse button is clicked, right is notified

CLICK.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

```
`javascript
```

```
function click(arg) {
//Obtains the clicked mouse button
var button = arg.button
}
、
```

Appearance

The appearance of the user handle can be customized by using the [size](#), [borderColor](#), [backgroundColor](#), [visible](#), [pathData](#), and [pathColor](#) properties of the [userHandles](#).

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                Shape = new { type = "Basic", shape = "Rectangle" },
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            List<DiagramUserHandle> Userhandle = new
List<DiagramUserHandle>();
            Userhandle.Add(new DiagramUserHandle()
            {
                Name = "clone",
                PathData = "M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3," +
                "0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z " +
                "M68.5,72.5h-30V34.4h30V72.5z",
                Visible = true,
                Offset = 0,
                Side = Side.Bottom,
                Margin = new DiagramMargin() { Left = 0, Right = 0, Top = 0,
Bottom = 0 }
            });
            ViewBag.userhandle = Userhandle;
            ViewBag.getTool = "getTool";
            return View();
        }
    }
}
```

```
`javascript
```

```
function getTool(action) {
var tool;
if (action === 'clone') {
tool = new CloneTool(diagram.commandHandler);
}
return tool;
}

var extends = (this && this.extends) || (function () {
var extendStatics = Object.setPrototypeOf ||
/ jshint proto: true /
({ proto: [] } instanceof Array && function (d, b) { d.proto = b; }) ||
function (d, b) { for (var p in b) if (b.hasOwnProperty(p)) d[p] = b[p]; };
return function (d, b) {
extendStatics(d, b);
function () { this.constructor = d; }
d.prototype = b === null ? Object.create(b) : (.prototype = b.prototype, new ());
};
})();
var CloneTool = (function (_super) {
extends(CloneTool, _super);
function CloneTool() {
return super !== null && super.apply(this, arguments) || this;
}
CloneTool.prototype.mouseDown = function (args) {
var newObject;
if (diagram.selectedItems.nodes.length > 0) {
newObject = ej.diagrams.cloneObject(diagram.selectedItems.nodes[0]);
}
else {
newObject = ej.diagrams.cloneObject(diagram.selectedItems.connectors[0]);
}
newObject.id += ej.diagrams.randomId();
diagram.paste([newObject]);
}
```



```

args.source = diagram.nodes[diagram.nodes.length - 1];
args.sourceWrapper = args.source.wrapper;
_super.prototype.mouseDown.call(this, args);
this.inAction = true;
};
return CloneTool;
}(ej.diagrams.MoveTool));
`

```

Zoom pan in diagram control

- When a large diagram is loaded, only certain portion of the diagram is visible. The remaining portions are clipped. Clipped portions can be explored by scrolling the scrollbars or panning the diagram.
- Diagram can be zoomed in or out by using Ctrl + mouse wheel.
- When the diagram is zoomed or panned, the [scrollChange](#) event gets triggered.



Zoom pan status

Diagram provides the support to notify the pan status of the zoom pan tool. Whenever the diagram is panning the [scrollChange](#) event is triggered and hence the pan status can be obtained. The pan status is notified with Start, Progress, and Completed.

Pan Status	Description
Start	When the mouse is clicked and dragged the status is notified as start.
Progress	When the mouse is in motion the status is notified as progress.
Completed	When panning is stopped the status is notified with completed.

PANSTATUS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
    }
}

```

```

public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
    nodes.Add(new DiagramNode() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        Shape = new { type = "Basic", shape = "Rectangle" },
        OffsetX = 100,
        OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    ViewBag.scrollChange = "scrollChange";
    return View();
}
}
}

```

```

`javascript

```

```

function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
diagram.tool = DiagramTools.ZoomPan;
diagram.dataBind();
}
function scrollChange(args) {
var panStatus = args.panState
}
,

```

Drag Resize and Rotate

Drag

- An object can be dragged by clicking and dragging it. When multiple elements are selected, dragging any one of the selected elements move every selected element.
- When you drag the elements in the diagram, the [positionChange](#) event gets triggered and to do customization in this event.



Resize

- Selector is surrounded by eight thumbs. When dragging these thumbs, selected items can be resized.
- When one corner of the selector is dragged, opposite corner is in a static position.
- When a node is resized, the [sizeChange](#) event gets triggered.



While dragging and resizing, the objects are snapped towards the nearest objects to make better alignments. For better alignments, refer to [Snapping](#).

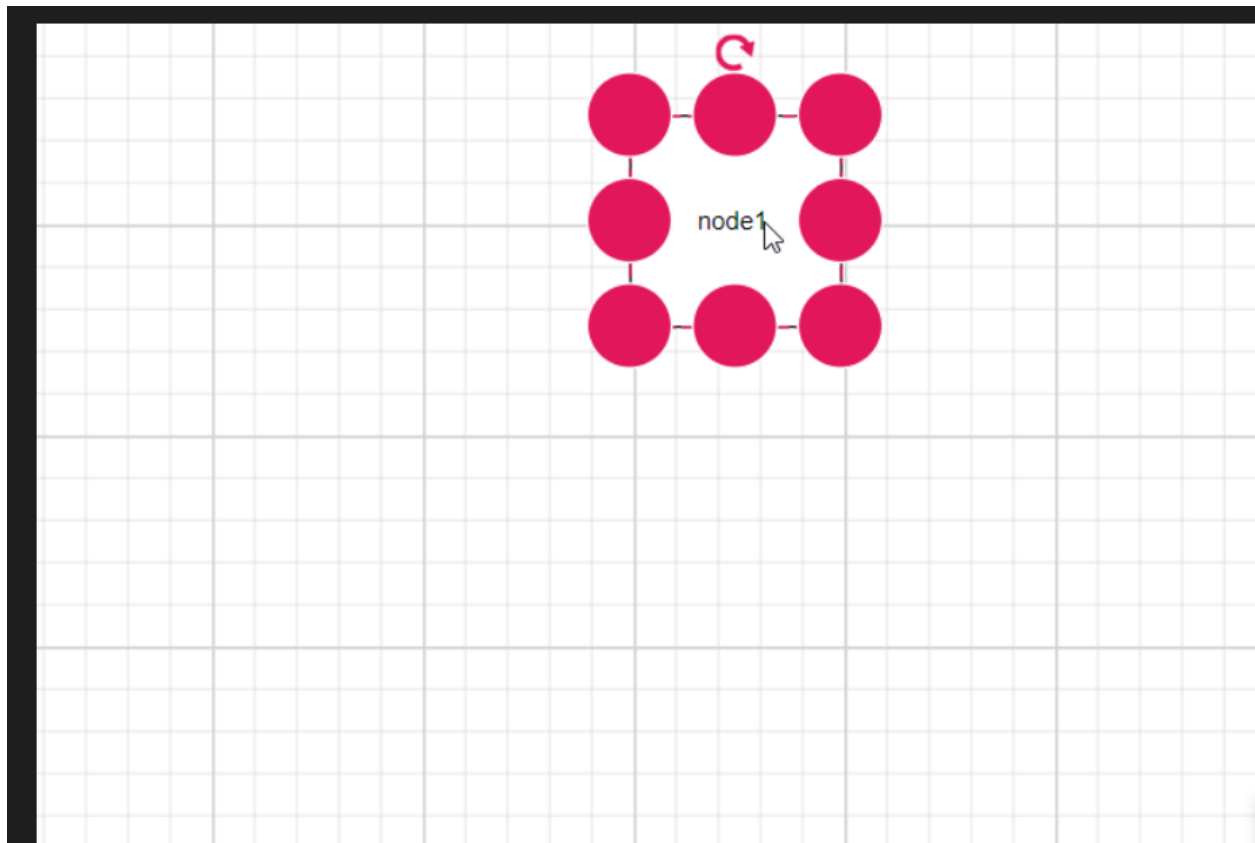
Customize the resize-thumb

You can change the size of the node resize thumb and the connector end point handle by using the `handleSize` property. The appearance such as fill, stroke, and stroke width of the node resize thumb and connector end point handle can be customized by overriding the `e-diagram-resize-handle` and `e-diagram-endpoint-handle` classes respectively.

NODE.CS

```
using System;  
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                BorderWidth=2,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Nodel
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```



Rotate

- A rotate handler is placed above the selector. Clicking and dragging the handler in a circular direction lead to rotate the node.
- The node is rotated with reference to the static pivot point.
- Pivot thumb (thumb at the middle of the node) appears while rotating the node to represent the static point.



Keyboard interactions

Diagram provides support to interact with the elements with key gestures. By default, some in-built commands are bound with a relevant set of key combinations.

The following table illustrates those commands with the associated key values.

Shortcut Key	Command	Description
-----	-----	-----
Ctrl + A	<code>selectAll</code>	Select all nodes/connectors in the diagram.
Ctrl + C	<code>copy</code>	Copy the diagram selected elements.
Ctrl + V	<code>paste</code>	Pastes the copied elements.
Ctrl + X	<code>cut</code>	Cuts the selected elements.
Ctrl + Z	<code>undo</code>	Reverses the last editing action performed on the diagram.
Ctrl + Y	<code>redo</code>	Restores the last editing action when no other actions have occurred since the last undo on the diagram.
Delete	<code>delete</code>	Deletes the selected elements.
Ctrl/Shift + Click on object		Multiple selection (Selector binds all selected nodes/connectors).
Up Arrow	<code>nudge("up")</code> <code>nudgeUp</code>	Moves the selected elements towards up by one pixel.
Down Arrow	<code>nudge("down")</code> <code>nudgeDown</code>	Moves the selected elements towards down by one pixel.
Left Arrow	<code>nudge("left")</code> <code>nudgeLeft</code>	Moves the selected elements towards left by one pixel.
Right Arrow	<code>nudge("right")</code> <code>nudgeRight</code>	Moves the selected elements towards right by one pixel.
Ctrl + MouseWheel	<code>zoom</code>	Zoom (Zoom in/Zoom out the diagram).
F2	<code>startLabelEditing</code>	Starts to edit the label of the selected element.
Esc	<code>endLabelEditing</code>	Sets the label mode as view and stops editing.

See Also

- [How to create diagram nodes using drawing tools](#)
- [How to create diagram connectors using drawing tools](#)
- [How to disable the diagram interaction](#)
- [How to control the diagram history](#)
- [How to create overview control to the diagram](#)

Tools in Diagram Control

Drawing tools

Drawing tool allows to draw any kind of node or connector during runtime by clicking and dragging on the diagram page.

Shapes

To draw a shape, set the JSON of that shape to the drawType property of the diagram and activate the drawing tool by using the [tool](#) property.

SHAPES.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

```
`javascript
```

```
function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var drawingshape = { type: 'Basic', shape: 'Rectangle' };
var node = {
shape: drawingshape
```

```

};
diagram.drawingObject = node;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}
`

```

SHAPES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Nodel
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

`javascript


```

function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var node = {
id: "Path",
style: { fill: "#fbe172" },
annotations: [{
content: "Path"
}],
shape: {
type:'Path',
data: 'M13.560 67.524 L 21.941 41.731 L 0.000 25.790 L 27.120 25.790 L 35.501 0.000 L 43.882 25.790 L
71.000 25.790 L 49.061 41.731 L 57.441 67.524 L 35.501 51.583 z'
}
};
diagram.drawingObject = node;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}

```

Connectors

To draw connectors, set the JSON of the connector to the drawType property. The drawing [tool](#) can be activated by using the tool property.

SHAPES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"

```

```

    }
    });
    nodes.Add(new Node() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}
}

```

```
`javascript
```

```

function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var connectors = {
id: 'connector1',
type: 'Straight',
segments: [{ type: "polyline" }]
}
diagram.drawingObject = connectors;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}
`

```

Text

Diagram allows to create a textNode, when you click on the diagram page.

NODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

```

```

using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            return View();
        }
        public class Node: DiagramNode {
            public string text;
        }
    }
}

```

```

`javascript

```

```

function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var node = {
shape: {
type: 'Text',
}
};
diagram.drawingObject = node;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();

```

```

}
,

```

Once you activate the TextTool, perform label editing of a node/connector.

Polygon shape

Diagram allows to create the polygon shape by clicking and moving the mouse at runtime on the diagram page.

NODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```

function Created() {
var drawingshape = { type: 'Basic', shape: 'Polygon' };

```

```
//JSON to create a polygon
var node = {
  shape: drawingshape
};
diagram.drawingObject = node;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}
、
```

Polyline Connector

Diagram allows to create the polyline segments with straight lines and angled vertices at the control points by clicking and moving the mouse at runtime on the diagram page.

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

```

    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```

function Created() {
var connector = { id: 'connector1', type: 'Polyline'};
diagram.drawingObject = connector;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}
`

```

Tool selection

There are some functionalities that can be achieved by clicking and dragging on the diagram surface. They are as follows.

- Draw selection rectangle: MultipleSelect tool
- Pan the diagram: Zoom pan
- Draw nodes/connectors: DrawOnce/DrawOnce

As all the three behaviors are completely different, you can achieve only one behavior at a time, based on the tool that you choose. When more than one of those tools are applied, a tool is activated based on the precedence given in the following table.

Precedence	Tools	Description
1st	ContinuesDraw	Allows you to draw the nodes or connectors continuously. Once it is activated, you cannot perform any other interaction in the diagram.
2nd	DrawOnce	Allows you to draw a single node or connector. Once you complete the DrawOnce action, SingleSelect, and MultipleSelect tools are automatically enabled.
3rd	ZoomPan	Allows you to pan the diagram. When you enable both the SingleSelect and ZoomPan tools, you can perform the basic interaction as the cursor hovers node/connector. Panning is enabled when cursor hovers the diagram.
4th	MultipleSelect	Allows you to select multiple nodes and connectors. When you enable both the MultipleSelect and ZoomPan tools, cursor hovers the diagram. When panning is enabled, you cannot select multiple nodes.
5th	SingleSelect	Allows you to select individual nodes or connectors.
6th	None	Disables all tools.

Set the desired [tool](#) to the tool property of the diagram model.

```
`javascript
function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var connectors = {
id: 'connector1',
type: 'Straight',
segments: [{ type: "Straight" }]
}
diagram.drawingObject = connectors;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce | DiagramTools.ZoomPan;
diagram.dataBind();
}
`
```

Events

SHAPES.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Nodel = new List <
DiagramNodeAnnotation > ();
            Nodel.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                text = "node1",
                OffsetX = 100,
```

```

        OffsetY = 100,
        Annotations = Node1
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```

function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var connectors = {
id: 'connector1',
type: 'Straight',
segments: [{ type: "Straight" }]
}
diagram.drawingObject = connectors;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}
`

```

Freehand Drawing

Diagram has support for free-hand drawing to draw anything on the diagram page independently. Free-hand drawing will be enabled by using the drawingObject property and setting its value to Freehand.

The following code illustrates how to draw a freehand drawing.

SHAPES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();

```



```

        List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
        Node1.Add(new DiagramNodeAnnotation() {
            Content = "node1", Style = new DiagramTextStyle() {
                Color = "White", StrokeColor = "None"
            }
        });
        nodes.Add(new Node() {
            Id = "node1",
            Width = 100,
            Height = 100,
            Style = new NodeStyleNodes() {
                Fill = "darkcyan"
            },
            text = "node1",
            OffsetX = 100,
            OffsetY = 100,
            Annotations = Node1
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```

function Created() {
var diagram = document.getElementById("container").ej2_instances[0];
var connectors = {
id: 'connector1',
type: 'Freehand',
}
diagram.drawingObject = connectors;
//To draw an object once, activate draw once
diagram.tool = DiagramTools.DrawOnce;
diagram.dataBind();
}
`

```

Gridlines

Gridlines are the pattern of lines drawn behind the diagram elements. It provides a visual guidance while dragging or arranging the objects on the diagram surface.

The model's [snapSettings](#) property is used to customize the gridlines and control the snapping behavior in the diagram.

Customize the gridlines visibility

The [snapSettings.snapConstraints](#) enables you to show or hide the gridlines.

If you need to enable snapping, then inject snapping module into the diagram.

SNAP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

To show only horizontal or vertical gridlines or to hide gridlines, refer to [Constraints](#).

Appearance

The appearance of the gridlines can be customized by using a set of predefined properties.

- The [horizontalGridLines](#) and the [verticalGridLines](#) properties allows to customize the appearance of the horizontal and vertical gridlines respectively.
- The horizontal gridlines [lineColor](#) and [lineDashArray](#) properties are used to customize the line color and line style of the horizontal gridlines.
- The vertical gridlines [lineColor](#) and [lineDashArray](#) properties are used to customize the line color and line style of the vertical gridlines.

APPEAR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            DiagramGridlines gridLines = new DiagramGridlines() {
                LineColor = "#e0e0e0", LineDashArray = "2,2"
            };
            ViewBag.gridLines = gridLines;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Line intervals

Thickness and the space between gridlines can be customized by using horizontal gridlines's [linesInterval](#) and vertical gridlines's [linesInterval](#) properties. In the lines interval collections, values at the odd places are referred as the thickness of lines and values at the even places are referred as the space between gridlines.

INTERVALS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {

```

```

// GET: Nodes
public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    nodes.Add(new Node() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "#6BA5D7",
            StrokeColor = "White"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    double[] intervals = { 1, 9, 0.25, 9.75, 0.25, 9.75, 0.25, 9.75,
0.25, 9.75, 0.25, 9.75, 0.25, 9.75, 0.25, 9.75, 0.25, 9.75 };
    DiagramGridlines gridLines = new DiagramGridlines() {
        LineColor = "#e0e0e0", LineIntervals = intervals,
        LineDashArray = "2,2"
    };
    ViewBag.gridLines = gridLines;
    return View();
}

public class Node: DiagramNode {
    public string text;
}
}

```

Snapping

Snap to lines

This feature allows the diagram objects to snap to the nearest intersection of gridlines while being dragged or resized. This feature enables easier alignment during layout or design.

Snapping to gridlines can be enabled or disabled with the [snapSettings.snapConstraints](#).

LINES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
            });
        }
    }
}

```

```

        Style = new NodeStyleNodes() {
            Fill = "#6BA5D7",
            StrokeColor = "White"
        },
        OffsetX = 100,
        OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    return View();
}

public class Node: DiagramNode {
    public string text;
}
}

```

Customization of snap intervals

By default, the objects are snapped towards the nearest gridline. The gridline or position towards where the diagram object snaps can be customized with the horizontal gridlines's [snapInterval](#) and the vertical gridlines's [snapInterval](#) properties.

SNAPINTERVALS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            double[] snap = { 10 };
            DiagramGridlines gridLines = new DiagramGridlines() {
                SnapIntervals = snap
            };
            ViewBag.gridLines = gridLines;
            return View();
        }
    }
}

```

```

public class Node: DiagramNode {
    public string text;
}

```

Snap to objects

The snap to object provides visual cues to assist with aligning and spacing diagram elements. A node can be snapped with its neighbouring objects based on certain alignments. Such alignments are visually represented as smart guides.

The [snapObjectDistance](#) property allows to define minimum distance between the selected object and the nearest object.

The [snapAngle](#) property allows to define the snap angle by which the object needs to be rotated.

The [snapConstraints](#) property allows to enable or disable the certain features of the snapping, refer to `snapConstraints`.

The [snapLineColor](#) property allows to define the color of the snapline.

OBJECTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Page Settings

Page settings can customize the appearance, width, and height of the diagram page.

Page size and appearance

- The size and appearance of the diagram pages can be customized with the page settings property.
- The [width](#) and [height](#) properties of page settings define the size of the page and based on the size, the [orientation](#) will be set for the page. In addition to that, the appearance of the page can be customized with [source](#) and set of appearance specific properties.
- The [color](#) property is used to customize the background color and border color of the page.
- The [margin](#) property is used to define the page margin.
- To explore those properties, refer to [Page Settings](#).

PAGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

Set background image

Stretch and align the background image anywhere over the diagram area. The [source](#) property of [background](#) allows to set the path of the image. The [scale](#) and the [align](#) properties help to stretch/align the background images.

CSHTML

```

<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%).Height("590px").GetNodeDefaults
("getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSetti
ngs(s => s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
        .SnapSettings(se =>
se.Constraints(Syncfusion.EJ2.Diagrams.SnapConstraints.None)).pageSettings(p
e => pe..Width("500px").Height("590px")).Render()
        </div>
    </div>

```

IMAGE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Multiple page and page breaks

When multiple page is enabled, the size of the page dynamically increases or decreases in multiples of page width and height and completely fits diagram within the page boundaries. Page breaks is used as a visual guide to see how pages are split into multiple pages.

The [multiplePage](#) and [showPageBreak](#) properties of page settings allows you to enable/disable multiple pages and page breaks respectively.

CSHTML

```
<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults(
    "getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSettings(s => s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
    .SnapSettings(se =>
se.Constraints(Syncfusion.EJ2.Diagrams.SnapConstraints.None)).pageSettings(pe => pe.Width("500px").Height("590px")).Render()
    </div>
</div>
```

IMAGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

Boundary constraints

The diagram provides support to restrict/customize the interactive region, out of which the elements cannot be dragged, resized, or rotated. The [boundaryConstraints](#) property of page settings allows you to customize the interactive region. To explore the boundary constraints, refer to [Boundary Constraints](#).

CSHTML

```
<div class="col-lg-12 control-section">
  <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults(
  "getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSettings(s => s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
  .SnapSettings(se =>
se.Constraints(Syncfusion.EJ2.Diagrams.SnapConstraints.None)).pageSettings(pe =>
pe.Width("500px").Height("590px").boundaryConstraints("Page").orientation("Landscape")).Render()
  </div>
</div>
```

BOUNDARY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}
```

Scroll Settings in Diagram

The diagram can be scrolled by using the vertical and horizontal scrollbars. In addition to the scrollbars, mousewheel can be used to scroll the diagram. Diagram's [scrollSettings](#) enables to read the current scroll status, view port size, current zoom, and zoom factor. It also allows to scroll the diagram programmatically.

Get current scroll status

Scroll settings allows to read the scroll status, [viewPortWidth](#), [viewPortHeight](#), and [currentZoom](#) with a set of properties. To explore those properties, see [Scroll Settings](#).

Define scroll status

Diagram allows to pan the diagram before loading, so that any desired region of a large diagram is made to view. You can programmatically pan the diagram with the [horizontalOffset](#) and [verticalOffset](#) properties of scroll settings.

CSHTML

```
<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults
("getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSetti
ngs(s => s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
    .scrollsettings(se =>
se.horizontalOffset("100").verticalOffset("50"))).Render()
    </div>
</div>
```

STATUS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
        }
    }
}
```

```

        return View();
    }
}
public class Node: DiagramNode {
    public string text;
}
}

```

Update scroll status

You can programmatically change the scroll offsets at runtime by using the client-side method update.

CSHTML

```

<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults
("getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSetti
ngs(s => s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
    ).Render()

    </div>
</div>

```

UPDATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
}

```

```
`javascript

```

```
var diagram = document.getElementById("container").ej2_instances[0];

```

```
//Updates scroll settings

```

```
diagram.scrollSettings.horizontalOffset=200;

```

```
diagram.scrollSettings.verticalOffset=30

```

```
diagram.dataBind();

```

```
`

```

AutoScroll

Autoscroll feature automatically scrolls the diagram, whenever the node or connector is moved beyond the boundary of the diagram. So that, it is always visible during dragging, resizing, and multiple selection operations. Autoscroll is automatically triggered when any one of the following is done towards the edges of the diagram.

- Node dragging, resizing
- Connection editing
- Rubber band selection
- Label dragging

The diagram client-side event [ScrollChange](#) gets triggered when the autoscroll (scrollbars) is changed and you can do your own customization in this event.

The autoscroll behavior in your diagram can be enabled or disabled by using the [canAutoScroll](#) property of the diagram.

CSHTML

```
<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").Nodes(ViewBag.n
odes.margin)GetNodeDefaults("getNodeDefaults").GetConnectorDefaults("getConn
ectorDefaults").ScrollSettings(s =>
s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
    .scrollsettings(se => se.autoScrollBorder("Infinity"))).Render()
    </div>
</div>

```

AUTOSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {

```

```

public partial class DiagramController: Controller {
    // GET: Nodes
    public ActionResult Nodes() {
        List < DiagramNode > nodes = new List < DiagramNode > ();
        nodes.Add(new Node() {
            Id = "node1",
            Width = 100,
            Height = 100,
            Style = new NodeStyleNodes() {
                Fill = "#6BA5D7",
                StrokeColor = "White"
            },
            text = "node1",
            OffsetX = 100,
            OffsetY = 100,
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

Autoscroll border

The autoscroll border is used to specify the maximum distance between the object and diagram edge to trigger autoscroll. The default value is set as 15 for all sides (left, right, top, and bottom) and it can be changed by using the [autoScrollBorder](#) property of page settings.

CSHTML

```

<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").Nodes(ViewBag.nodes.margin)
getNodeDefaults("getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSettings(s =>
s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
.scrollsettings(se => se.autoScrollBorder("Infinity"))).Render()

    </div>
</div>

```

AUTO.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes

```

```

public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    nodes.Add(new Node() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "#6BA5D7",
            StrokeColor = "White"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    List<DiagramMargin> margin = new List<DiagramMargin>();
    margin.Add(new DiagramMargin() { Left = 100, Right = 100, Top =
100, Bottom = 100 });
    ViewBag.margin = margin;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

Scroll limit

The scroll limit allows to define the scrollable region of the diagram. It includes the following options:

- Allows to scroll in all directions without any restriction.
- Allows to scroll within the diagram content.
- Allows to scroll within the specified scrollable area.
- The [scrollLimit](#) property of scroll settings helps to limit the scrolling.

The scrollSettings [scrollableArea](#) allows to extend the scrollable region that is based on the scroll limit.

CSHTML

```

<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").Nodes(ViewBag.n
odes.margin)GetNodeDefaults("getNodeDefaults").GetConnectorDefaults("getConn
ectorDefaults").ScrollSettings(s =>
s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
        .scrollsettings(se => se.autoScrollBorder("Infinity"))).Render()
    </div>
</div>

```

DEFAULT.CS

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

Scroll padding

The scroll padding allows to extend the scrollable region that is based on the scroll limit.

CSHTML

```

<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").Nodes(ViewBag.n
odes.margin)GetNodeDefaults("getNodeDefaults").GetConnectorDefaults("getConn
ectorDefaults").ScrollSettings(s => s.Padding(new DiagramMargin() { Right =
50, Bottom = 50})).Render()
    </div>
</div>

```

DEFAULT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {

```



```

public partial class DiagramController: Controller {
    // GET: Nodes
    public ActionResult Nodes() {
        List < DiagramNode > nodes = new List < DiagramNode > ();
        nodes.Add(new Node() {
            Id = "node1",
            text = "node1",
            Offset = 0,
            side = "Right",
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

Scrollable Area

Scrolling beyond any particular rectangular area can be restricted by using the [scrollableArea](#) property of scroll settings. To restrict scrolling beyond any custom region, set the [scrollLimit](#) as “limited”.

CSHTML

```

<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults(
    "getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSettings(s => s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity))
    .scrollsettings(se => se.scrollLimit("Infinity")).Render()

    </div>
</div>

```

AREA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",

```

```

        StrokeColor = "White"
    },
    text = "node1",
    OffsetX = 100,
    OffsetY = 100,
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

UpdateViewport

The [updateViewPort](#) method is used to update the diagram page and view size at runtime.

Data Binding in Diagram

- Diagram can be populated with the **nodes** and **connectors** based on the information provided from an external data source.
- Diagram exposes its specific data-related properties allowing you to specify the data source fields from where the node information has to be retrieved from.
- The [dataManager](#) property is used to define the data source either as a collection of objects or as an instance of **DataManager** that needs to be populated in the diagram.
- The [ID](#) property is used to define the unique field of each JSON data.
- The [parentId](#) property is used to define the parent field which builds the relationship between ID and parent field.
- The [root](#) property is used to define the root node for the diagram populated from the data source.
- To explore those properties, see [DataSourceSettings](#).
- Diagram supports two types of data binding. They are:
 1. Local data
 2. Remote data

Local data

Diagram can be populated based on the user defined JSON data (Local Data) by mapping the relevant data source fields.

To map the user defined JSON data with diagram, configure the fields of [dataSourceSettings](#).

LOCAL.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
    }
}

```

```

{
    // GET: LocalData
    public IActionResult LocalData()
    {
        List<LocalDataDetails> localData = new List<LocalDataDetails>();
        localData.Add(new LocalDataDetails("Species", ""));
        localData.Add(new LocalDataDetails("Plants", "Species"));
        localData.Add(new LocalDataDetails("Fungi", "Species"));
        localData.Add(new LocalDataDetails("Lichens", "Species"));
        localData.Add(new LocalDataDetails("Animals", "Species"));
        localData.Add(new LocalDataDetails("Mosses", "Plants"));
        localData.Add(new LocalDataDetails("Ferns", "Plants"));
        localData.Add(new LocalDataDetails("Gymnosperms", "Plants"));
        localData.Add(new LocalDataDetails("Dicotyledens", "Plants"));
        localData.Add(new LocalDataDetails("Monocotyledens", "Plants"));
        localData.Add(new LocalDataDetails("Invertebrates", "Animals"));
        localData.Add(new LocalDataDetails("Vertebrates", "Animals"));
        localData.Add(new LocalDataDetails("Insects", "Invertebrates"));
        localData.Add(new LocalDataDetails("Molluscs",
        "Invertebrates"));
        localData.Add(new LocalDataDetails("Crustaceans",
        "Invertebrates"));
        localData.Add(new LocalDataDetails("Others", "Invertebrates"));
        localData.Add(new LocalDataDetails("Fish", "Vertebrates"));
        localData.Add(new LocalDataDetails("Amphibians",
        "Vertebrates"));
        localData.Add(new LocalDataDetails("Reptiles", "Vertebrates"));
        localData.Add(new LocalDataDetails("Birds", "Vertebrates"));
        localData.Add(new LocalDataDetails("Mammals", "Vertebrates"));
        ViewBag.Nodes = localData;
        ViewBag.getNodeDefaults = "getNodeDefaults";
        ViewBag.getConnectorDefaults = "getConnectorDefaults";
        return View();
    }
}

public class LocalDataDetails
{
    public string Name { get; set; }
    public string Category { get; set; }

    public LocalDataDetails(string id, string category)
    {
        this.Name = id;
        this.Category = category;
    }
}
}

```

```

`javascript

```

```

function diagramCreated() {
    var diagram = document.getElementById("diagram").ej2_instances[0];
    diagram.tool = ej.diagrams.DiagramTools.ZoomPan;
    diagram.dataBind();
}

```

```
}  
function getNodeDefaults(obj, data, diagram) {  
  //Initialize shape  
  obj.annotations = [{  
    / tslint:disable:no-string-literal /  
    content: obj.data.Name, margin: { top: 10, left: 10, right: 10, bottom: 0 },  
    style: { color: 'black' }  
  }];  
  obj.style =  
  {  
    fill: '#ffeec7', strokeColor: '#f5d897', strokeWidth: 1  
  }  
  ; obj.shape = {  
    type: 'Basic', shape: 'Rectangle'  
  }  
  ; obj.width = 95; obj.height = 30; return obj;  
}  
function getConnectorDefaults(connector, diagram) {  
  connector.type = 'Orthogonal';  
  connector.style.strokeColor = '#4d4d4d';  
  connector.targetDecorator.shape = 'None';  
  return connector;  
}  
`
```

CRUD

This feature allows to read the data source and perform add or edit or delete the data in data source at runtime.

Read DataSource

- This feature allows to define the nodes and connectors collection in the data source and `connectionDataSource` respectively.
- You can set the data collection in the model's `dataSourceSettings` [dataManager](#) property. The nodes will be generated based on the data specified in the data source.
- You can set the connector collection in the model's `dataSourceSettings` [connectionDataSource](#) property.

- The dataSourceSettings connectionDataSource [dataManager](#) property is used to set the data source for the connection data source items.
- If you have a data (data will be set in the dataSource property) with parent relationship in the database and also defined the connector in the connectionDataSource simultaneously, then the connectors set in the connectionDataSource will be considered as a priority to render the connector.
- The dataSourceSettings [crudAction's read](#) property specifies the method, which is used to read the data source and it populates the nodes in the diagram.
- The connectionDataSource crudAction's [read](#) specifies the method, which is used to read the data source and it populates the connectors in the diagram.
- The dataSourceSettings's [id](#) and connectionDataSource's [id](#) properties are used to define the unique field of each JSON data.
- The connectionDataSource's [sourceID](#) and [targetID](#) properties are used to set the sourceID and targetID for connection data source item.
- The connectionDataSource's [sourcePointX](#), [sourcePointY](#), [targetPointX](#), and [targetPointY](#) properties are used to define the sourcePoint and targetPoint values for connector from data source.
- The dataSourceSettings crudAction's [customFields](#) property is used to maintain the additional information for nodes.
- Similarly, connectionDataSource's crudAction's [customFields](#) is used to maintain the additional information for connectors.

How to perform Editing at runtime

- The dataSourceSettings crudAction object allows to define the method, which is used to get the changes done in the data source defined for shapes from the client-side to the server-side.
- Similarly, the connectionDataSource crudAction object allows to define the method, which is used to get the changes done in the data source defined for connectors from the client-side to the server-side.

InsertData

- The dataSourceSettings crudAction's [create](#) property specifies the method, which is used to get the nodes added from the client-side to the server-side.
- The connectionDataSource crudAction's [create](#) specifies the method, which is used to get the connectors added from the client-side to the server-side.

INSERTDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        // GET: LocalData
        public IActionResult LocalData()
```

```

        {
            CRUDAction nodeCrud = new CRUDAction()
            {
                //Define URL to perform CRUD operations with nodes records
                in database.
                Create =
                "https://js.syncfusion.com/demos/ejServices/api/Diagram/AddNodes",
            };
            ViewBag.NodeCrud = nodeCrud;
            ConnectionDataSource dataSource = new ConnectionDataSource()
            {
                Id = "Name",
                SourceID = "SourceNode",
                TargetID = "TargetNode",
                CrudAction = new CRUDAction()
                {
                    //Define URL to perform CRUD operations with connector
                    records in database.
                    Create =
                    "https://js.syncfusion.com/demos/ejServices/api/Diagram/AddConnectors",
                }
            };
            ViewBag.DataSource = dataSource;
            return View();
        }
    }

    public class CRUDAction
    {
        [DefaultValue(null)]
        [HtmlAttributeName("read")]
        [JsonProperty("read")]
        public string Read { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("create")]
        [JsonProperty("create")]
        public string Create { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("update")]
        [JsonProperty("update")]
        public string Update { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("destroy")]
        [JsonProperty("destroy")]
        public string Destroy { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("customFields")]
        [JsonProperty("customFields")]
        public object[] CustomFields { get; set; }
    }

    public class ConnectionDataSource
    {
        [DefaultValue(null)]
        [HtmlAttributeName("id")]
        [JsonProperty("id")]
        public string Id { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("sourceID")]

```

```

        [JsonProperty("sourceID")]
        public string SourceID { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("targetID")]
        [JsonProperty("targetID")]
        public string TargetID { get; set; }
        [DefaultValue(null)]
        [HtmlAttributeName("crudAction")]
        [JsonProperty("crudAction")]
        public CRUDAction CrudAction { get; set; }
    }
}

```

```
`javascript
```

```
var diagramElement = document.getElementById('element');
```

```
var diagram = diagramElement.ej2_instances[0];
```

//Sends the newly added nodes/connectors from client side to the server side through the URL which is specified in server side.

```
diagram.insertData();
```

```
,
```

UpdateData

- The dataSourceSettings crudAction's [update](#) property specifies the method, which is used to get the modified nodes from the client-side to the server-side.
- The connectionDataSource crudAction's [update](#) specifies the method, which is used to get the modified connectors from the client-side to the server-side.

UPDATEDATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        // GET: LocalData
        public IActionResult LocalData()
        {
            CRUDAction nodeCrud = new CRUDAction()
            {
                //Define URL to perform CRUD operations with nodes records
                //in database.
                Update =
                "https://js.syncfusion.com/demos/ejServices/api/Diagram/UpdateNodes",
            };
            ViewBag.NodeCrud = nodeCrud;
            ConnectionDataSource dataSource = new ConnectionDataSource()

```

```

        {
            Id = "Name",
            SourceID = "SourceNode",
            TargetID = "TargetNode",
            CrudAction = new CRUDAction()
            {
                //Define URL to perform CRUD operations with connector
                records in database.
                Update =
                "https://js.syncfusion.com/demos/ejServices/api/Diagram/UpdateConnectors",
            }
        };
        ViewBag.DataSource = dataSource;
        return View();
    }
}

public class CRUDAction
{
    [DefaultValue(null)]
    [HtmlAttributeName("read")]
    [JsonProperty("read")]
    public string Read { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("create")]
    [JsonProperty("create")]
    public string Create { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("update")]
    [JsonProperty("update")]
    public string Update { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("destroy")]
    [JsonProperty("destroy")]
    public string Destroy { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("customFields")]
    [JsonProperty("customFields")]
    public object[] CustomFields { get; set; }
}

public class ConnectionDataSource
{
    [DefaultValue(null)]
    [HtmlAttributeName("id")]
    [JsonProperty("id")]
    public string Id { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("sourceID")]
    [JsonProperty("sourceID")]
    public string SourceID { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("targetID")]
    [JsonProperty("targetID")]
    public string TargetID { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("crudAction")]
    [JsonProperty("crudAction")]
    public CRUDAction CrudAction { get; set; }
}

```



```
}
}
```

```
`javascript
```

```
var diagramElement = document.getElementById('element');
```

```
var diagram = diagramElement.ej2_instances[0];
```

```
//Sends the updated nodes/connectors from client side to the server side through the URL which is specified in server side.
```

```
diagram.updateData();
```

```
,
```

DeleteData

- The dataSourceSettings crudAction's [destroy](#) property specifies the method, which is used to get the deleted nodes from the client-side to the server-side.
- The connectionDataSource crudAction's [destroy](#) specifies the method, which is used to get the deleted connectors from the client-side to the server-side.

DELETEDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Diagram
{
    public partial class DiagramController : Controller
    {
        // GET: LocalData
        public IActionResult LocalData()
        {
            CRUDAction nodeCrud = new CRUDAction()
            {
                //Define URL to perform CRUD operations with nodes records
                in database.
                Destroy =
                "https://js.syncfusion.com/demos/ejServices/api/Diagram/DeleteNodes",
            };
            ViewBag.NodeCrud = nodeCrud;
            ConnectionDataSource dataSource = new ConnectionDataSource()
            {
                Id = "Name",
                SourceID = "SourceNode",
                TargetID = "TargetNode",
                CrudAction = new CRUDAction()
                {
                    //Define URL to perform CRUD operations with connector
                    records in database.
                    Destroy =
                    "https://js.syncfusion.com/demos/ejServices/api/Diagram/DeleteConnectors",
```

```

    }
    };
    ViewBag.DataSource = dataSource;
    return View();
}
}
}
public class CRUDAction
{
    [DefaultValue(null)]
    [HtmlAttributeName("read")]
    [JsonProperty("read")]
    public string Read { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("create")]
    [JsonProperty("create")]
    public string Create { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("update")]
    [JsonProperty("update")]
    public string Update { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("destroy")]
    [JsonProperty("destroy")]
    public string Destroy { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("customFields")]
    [JsonProperty("customFields")]
    public object[] CustomFields { get; set; }
}
public class ConnectionDataSource
{
    [DefaultValue(null)]
    [HtmlAttributeName("id")]
    [JsonProperty("id")]
    public string Id { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("sourceID")]
    [JsonProperty("sourceID")]
    public string SourceID { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("targetID")]
    [JsonProperty("targetID")]
    public string TargetID { get; set; }
    [DefaultValue(null)]
    [HtmlAttributeName("crudAction")]
    [JsonProperty("crudAction")]
    public CRUDAction CrudAction { get; set; }
}
}

```

```
`javascript
```

```
var diagramElement = document.getElementById('element');
```

```
var diagram = diagramElement.ej2_instances[0];
```

//Sends the deleted nodes/connectors from client side to the server side through the URL which is specified in server side.

```
diagram.removeData();
```

,

See Also

- [How to arrange the diagram nodes and connectors using varies layout](#)

Automatic Layout in Diagram

Diagram provides support to auto-arrange the nodes in the diagram area that is referred as [Layout](#). It includes the following layout modes:

Layout modes

- Hierarchical layout
- Organization chart
- Radial tree
- Symmetric layout
- Mind Map layout
- Complex hierarchical tree layout

Hierarchical layout

The hierarchical tree layout arranges nodes in a tree-like structure, where the nodes in the hierarchical layout may have multiple parents. There is no need to specify the layout root. To arrange the nodes in a hierarchical structure, specify the layout [type](#) as `HierarchicalTree`.

HIERARCHICAL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };
            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class HierarchicalDetails
```

```

{
    public string Name { get; set; }
    public string Category { get; set; }
    public string FillColor { get; set; }
    public HierarchicalDetails(string name, string category, string
fillcolor)
    {
        this.Name = name;
        this.Category = category;
        this.FillColor = fillcolor;
    }
    public static List<HierarchicalDetails> GetData()
    {
        List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
        hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
        return hierarchicaldetails;
    }
}
}

```

```
`javascript
```

```

function getNodeDefaults(obj, diagram) {
var collapseicon = obj.collapseicon;
obj.shape = { type: 'Text', content: obj.data.Name };
obj.style = { fill: '#659be5', strokeColor: 'none', color: 'white', strokeWidth: 2 };
obj.borderColor = '#3a6b5';
obj.backgroundColor = '#659be5';

```

```

obj.shape.margin = { left: 5, right: 5, bottom: 5, top: 5 };
obj.expandIcon = { height: 10, width: 10, shape: 'None', fill: 'lightgray', offset: { x: .5, y: 1 } };
obj.expandIcon.verticalAlignment = 'Auto';
obj.expandIcon.margin = { left: 0, right: 0, top: 0, bottom: 0 };
collapseIcon.offset = { x: .5, y: 1 };
collapseIcon.verticalAlignment = 'Auto';
collapseIcon.margin = { left: 0, right: 0, top: 0, bottom: 0 };
collapseIcon.height = 10;
collapseIcon.width = 10;
collapseIcon.padding.top = 5;
collapseIcon.shape = 'None';
collapseIcon.fill = 'lightgray';
return obj;
}

function connectorDefaults(connector, diagram) {
connector.targetDecorator.shape = 'None';
connector.type = 'Orthogonal';
connector.constraints = 0;
connector.cornerRadius = 5;
return connector;
}
`

```

Radial tree layout

The radial tree layout arranges nodes on a virtual concentric circle around a root node. Sub-trees formed by the branching of child nodes are located radially around the child nodes. This arrangement results in an ever-expanding concentric arrangement with radial proximity to the root node indicating the node level in the hierarchy. The layout [root](#) property can be used to define the root node of the layout. When no root node is set, the algorithm automatically considers one of the diagram nodes as the root node.

To arrange nodes in a radial tree structure, set the [type](#) of the layout as **RadialTree**.

RADIAL.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;

```

```

namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = RadialTreeDetails.GetData();
            return View();
        }
    }
    public class RadialTreeDetails
    {
        public string Id { get; set; }
        public string Name { get; set; }
        public string Designation { get; set; }
        public string ReportingPerson { get; set; }
        public RadialTreeDetails(string id, string name, string designation,
string reportingPerson)
        {
            this.Id = id;
            this.Name = name;
            this.Designation = designation;
            this.ReportingPerson = reportingPerson;
        }
        public static List<RadialTreeDetails> GetData()
        {
            List<RadialTreeDetails> radialTreeDetails = new
List<RadialTreeDetails>();
            radialTreeDetails.Add(new RadialTreeDetails("parent", "Maria
Anders", "Managing Director", ""));
            radialTreeDetails.Add(new RadialTreeDetails("1", "Ana Trujillo",
"Project Manager", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("2", "Lino Rodri",
"Project Manager", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("3", "Philip
Cramer", "Project Manager", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("4", "Pedro Afonso",
"Project Manager", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("5", "Anto Moreno",
"Project Lead", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("6", "Elizabeth
Roel", "Project Lead", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("7", "Aria Cruz",
"Project Lead", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("8", "Eduardo Roel",
"Project Lead", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("9", "Howard Snyder",
"Project Lead", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("10", "Daniel
Tonini", "Project Lead", "1"));
            radialTreeDetails.Add(new RadialTreeDetails("11", "Nardo
Batista", "Project Lead", "1"));
            return radialTreeDetails;
        }
    }
}

```

```
}

```

Organizational Chart

An organizational chart is a diagram that displays the structure of an organization and relationships. To create an organizational chart, the [type](#) of layout should be set as an **OrganizationalChart**.

ORGANIZATIONAL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = OrganizationalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            ViewBag.getLayoutInfo = "getLayoutInfo";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
};
            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class OrganizationalDetails
    {
        public string Id { get; set; }
        public string Role { get; set; }
        public string Color { get; set; }
        public string Manager { get; set; }
        public string ChartType { get; set; }
        public OrganizationalDetails(string id, string role, string color,
string manager, string chartType)
        {
            this.Id = id;
            this.Role = role;
            this.Color = color;
            this.Manager = manager;
            this.ChartType = chartType;
        }
        public static List<OrganizationalDetails> GetData()
        {
            List<OrganizationalDetails> organizationaldetails = new
List<OrganizationalDetails>();
            organizationaldetails.Add(new OrganizationalDetails("parent",
"Board", "#71AF17", "", ""));
            organizationaldetails.Add(new OrganizationalDetails("1",
"General Manager", "#71AF17", "parent", "right"));

```

```

        organizationaldetails.Add(new OrganizationalDetails("11",
"Assistant General Manager", "#71AF17", "1", ""));
        organizationaldetails.Add(new OrganizationalDetails("2", "Human
Resource Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("3",
"Trainers", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("4",
"Recruiting Team", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("6", "Design
Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("7", "Design
Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("8",
"Development Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("9",
"Drafting Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("10",
"Operations Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Statistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("12",
"Logistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("16",
"Marketing Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("17",
"Oversea sales Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("18",
"Petroleum Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("20",
"Service Dept. Manager", "#2E95D8", "10", ""));
        return organizationaldetails;
    }
}
}

```

```
`javascript
```

```

function getLayoutInfo(node, options, orientation, type) {
if (node.data.Role === 'General Manager') {
options.assistants.push(options.children[0]);
options.children.splice(0, 1);
}
if (!options.hasSubTree) {
options.type = 'Right';
}
}
`

```

Organizational chart layout starts parsing from root and iterate through all its child elements. The `getLayoutInfo` method provides necessary information of a node's children and the way to arrange

(direction, orientation, offsets, etc.) them. The arrangements can be customized by overriding this function as explained.

[GetLayoutInfo](#)

Set chart orientations, chart types, and offset to be left between parent and child nodes by overriding the method, `diagram.layout.getLayoutInfo`. The [getLayoutInfo](#) method is called to configure every subtree of the organizational chart. It takes the following arguments.

- node: Parent node to that options are to be customized.
- options: Object to set the customizable properties.

ORGANIZATIONAL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = OrganizationalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            ViewBag.getLayoutInfo = "getLayoutInfo";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };

            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class OrganizationalDetails
    {
        public string Id { get; set; }
        public string Role { get; set; }
        public string Color { get; set; }
        public string Manager { get; set; }
        public string ChartType { get; set; }
        public OrganizationalDetails(string id, string role, string color,
string manager, string chartType)
        {
            this.Id = id;
            this.Role = role;
            this.Color = color;
            this.Manager = manager;
            this.ChartType = chartType;
        }
        public static List<OrganizationalDetails> GetData()
        {
```

```

        List<OrganizationalDetails> organizationaldetails = new
List<OrganizationalDetails>();
        organizationaldetails.Add(new OrganizationalDetails("parent",
"Board", "#71AF17", "", ""));
        organizationaldetails.Add(new OrganizationalDetails("1",
"General Manager", "#71AF17", "parent", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Assistant General Manager", "#71AF17", "1", ""));
        organizationaldetails.Add(new OrganizationalDetails("2", "Human
Resource Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("3",
"Trainers", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("4",
"Recruiting Team", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("6", "Design
Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("7", "Design
Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("8",
"Development Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("9",
"Drafting Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("10",
"Operations Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Statistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("12",
"Logistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("16",
"Marketing Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("17",
"Oversea sales Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("18",
"Petroleum Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("20",
"Service Dept. Manager", "#2E95D8", "10", ""));
        return organizationaldetails;
    }
}
}

```

```
`javascript
```

```

function getLayoutInfo(node, options, orientation, type) {
if (!options.hasSubTree) {
options.type = 'Center';
options.orientation = 'Horizontal';
}
}
`

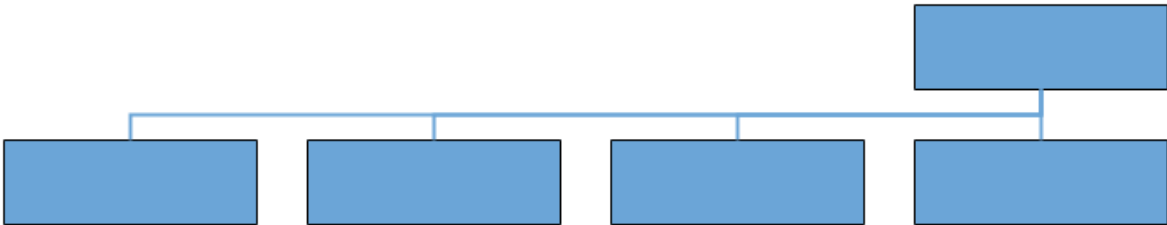
```

The following table illustrates the properties that “options” argument takes.

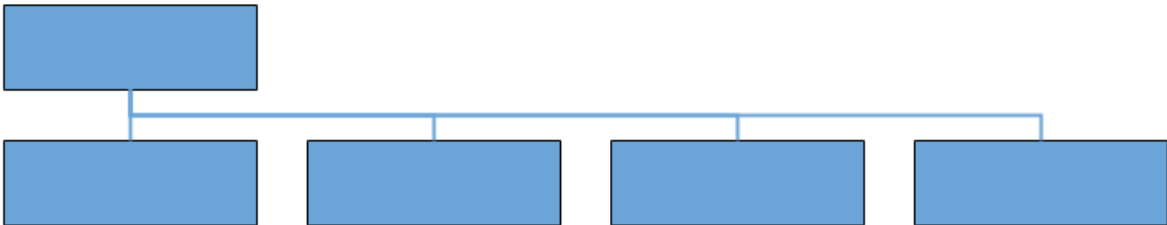
Property	Description	Default Value
options.assistants	By default, the collection is empty. When any of the child nodes have to be set as Assistant , you can remove from children collection and have to insert into assistants collection.	Empty array
options.orientation	Gets or sets the organizational chart orientation.	SubTreeOrientation.Vertical
options.type	Gets or sets the chart organizational chart type. For horizontal chart orientation:SubTreeAlignments.Center and for vertical chart orientation:SubTreeAlignments.Alternate	
options.offset	Offset is the horizontal space to be left between parent and child nodes.	20 pixels applicable only for vertical chart orientations.
options.hasSubTree	Gets whether the node contains subtrees.	Boolean
options.level	Gets the depth of the node from layout root.	Number
options.enableRouting	By default, connections are routed based on the chart type and orientations. This property gets or sets whether default routing is to be enabled or disabled.	true
options.rows	Sets the number of rows on which the child nodes will be arranged. Applicable only for balanced type horizontal tree.	Number

The following table illustrates the different chart orientations and chart types.

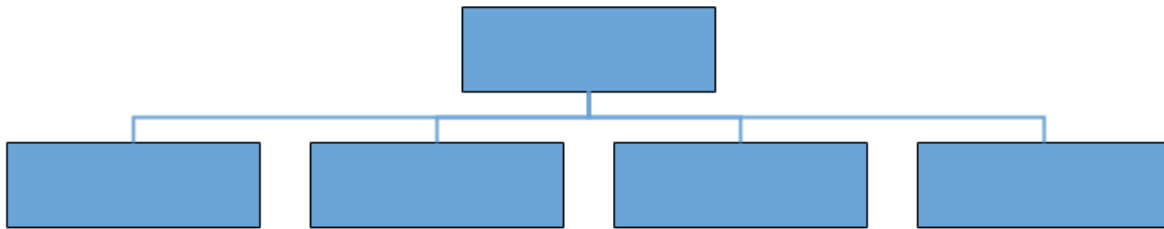
Orientation	Type	Description	Example
Horizontal	Left	Arranges the child nodes horizontally at the left side of the parent.	



Right	Arranges the child nodes horizontally at the right side of the parent.
-------	--

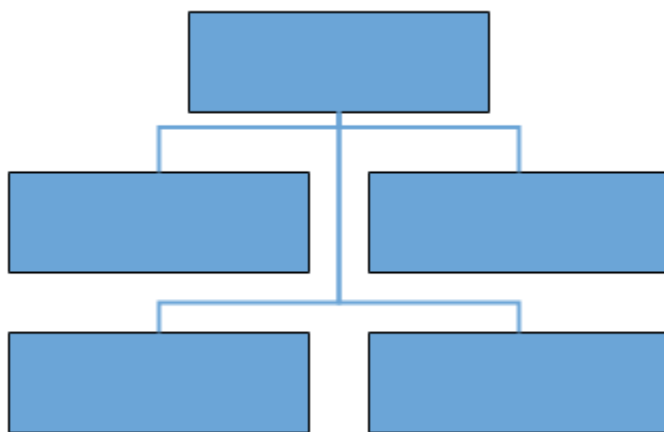


||Center|Arranges the children like standard tree layout orientation.|



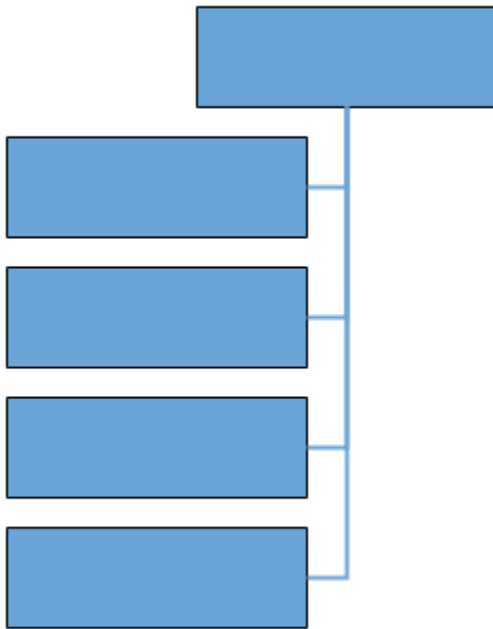
|

||Balanced|Arranges the leaf level child nodes in multiple rows.|



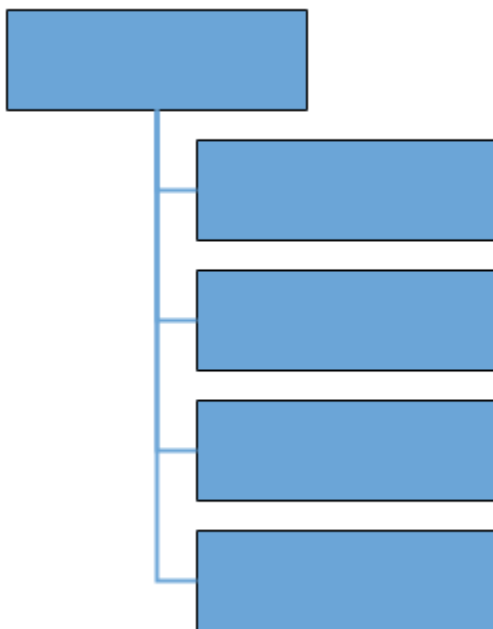
|

|Vertical|Left|Arranges the children vertically at the left side of the parent. |



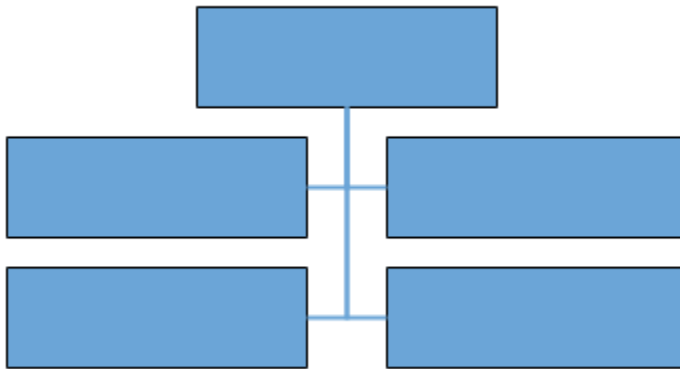
|

||Right|Arranges the children vertically at the right side of the parent. |



|

| | Alternate | Arranges the children vertically at both left and right sides of the parent. |



ORGANIZATIONAL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = OrganizationalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            ViewBag.getLayoutInfo = "getLayoutInfo";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };

            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class OrganizationalDetails
    {
        public string Id { get; set; }
        public string Role { get; set; }
        public string Color { get; set; }
        public string Manager { get; set; }
        public string ChartType { get; set; }
        public OrganizationalDetails(string id, string role, string color,
string manager, string chartType)
        {
            this.Id = id;
            this.Role = role;
            this.Color = color;
            this.Manager = manager;
        }
    }
}
```

```

        this.ChartType = chartType;
    }
    public static List<OrganizationalDetails> GetData()
    {
        List<OrganizationalDetails> organizationaldetails = new
List<OrganizationalDetails>();
        organizationaldetails.Add(new OrganizationalDetails("parent",
"Board", "#71AF17", "", ""));
        organizationaldetails.Add(new OrganizationalDetails("1",
"General Manager", "#71AF17", "parent", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Assistant General Manager", "#71AF17", "1", ""));
        organizationaldetails.Add(new OrganizationalDetails("2", "Human
Resource Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("3",
"Trainers", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("4",
"Recruiting Team", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("6", "Design
Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("7", "Design
Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("8",
"Development Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("9",
"Drafting Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("10",
"Operations Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Statistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("12",
"Logistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("16",
"Marketing Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("17",
"Oversea sales Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("18",
"Petroleum Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("20",
"Service Dept. Manager", "#2E95D8", "10", ""));
        return organizationaldetails;
    }
}
}

```

```

`javascript

```

```

function getLayoutInfo(node, options, orientation, type) {
if (!options.hasSubTree) {
options.type = 'Right';
options.orientation = 'Vertical';
}
}

```

```
}
,
```

Assistant

Assistants are child item that have a different relationship with the parent node. They are laid out in a dedicated part of the tree. A node can be specified as an assistant of its parent by adding it to the `assistants` property of the argument "options".

ORGANIZATIONAL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = OrganizationalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            ViewBag.getLayoutInfo = "getLayoutInfo";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };

            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class OrganizationalDetails
    {
        public string Id { get; set; }
        public string Role { get; set; }
        public string Color { get; set; }
        public string Manager { get; set; }
        public string ChartType { get; set; }
        public OrganizationalDetails(string id, string role, string color,
string manager, string chartType)
        {
            this.Id = id;
            this.Role = role;
            this.Color = color;
            this.Manager = manager;
            this.ChartType = chartType;
        }
        public static List<OrganizationalDetails> GetData()
        {
            List<OrganizationalDetails> organizationaldetails = new
List<OrganizationalDetails>();
            organizationaldetails.Add(new OrganizationalDetails("parent",
"Board", "#71AF17", "", ""));
```



```

        organizationaldetails.Add(new OrganizationalDetails("1",
"General Manager", "#71AF17", "parent", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Assistant General Manager", "#71AF17", "1", ""));
        organizationaldetails.Add(new OrganizationalDetails("2", "Human
Resource Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("3",
"Trainers", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("4",
"Recruiting Team", "#2E95D8", "2", ""));
        organizationaldetails.Add(new OrganizationalDetails("6", "Design
Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("7", "Design
Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("8",
"Development Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("9",
"Drafting Supervisor", "#2E95D8", "6", ""));
        organizationaldetails.Add(new OrganizationalDetails("10",
"Operations Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("11",
"Statistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("12",
"Logistics Department", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("16",
"Marketing Manager", "#1859B7", "1", "right"));
        organizationaldetails.Add(new OrganizationalDetails("17",
"Oversea sales Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("18",
"Petroleum Manager", "#2E95D8", "10", ""));
        organizationaldetails.Add(new OrganizationalDetails("20",
"Service Dept. Manager", "#2E95D8", "10", ""));
        return organizationaldetails;
    }
}
}

```

```
`javascript
```

```

function getLayoutInfo(node, options, orientation, type) {
if (node.data['Role'] === 'General Manager') {
options.assistants.push(options.children[0]);
options.children.splice(0, 1);
}
if (!options.hasSubTree) {
options.type = 'Right';
options.orientation = 'Vertical';
}
}

```

Symmetric layout

The symmetric layout has been formed using nodes position by closer together or pushing them further apart. This is repeated iteratively until the system comes to an equilibrium state.

The layout's [springLength](#) defined as how long edges should be, ideally. This will be the resting length for the springs. Edge attraction and vertex repulsion forces to be defined by using layout's [springFactor](#), the more sibling nodes repel each other. The relative positions do not change any more from one iteration to the next. The number of iterations can be specified by using layout's [maxIteration](#).

SYMMETRIC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = SymmetricalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            return View();
        }
    }
    public class SymmetricalDetails
    {
        public string Id { get; set; }
        public string Source { get; set; }
        public string Type { get; set; }
        public string ReportingPerson { get; set; }
        public SymmetricalDetails(string id, string source, string type)
        {
            this.Id = id;
            this.Source = source;
            this.Type = type;
        }
        public static List<SymmetricalDetails> GetData()
        {
            List<SymmetricalDetails> symmetricalDetails = new
            List<SymmetricalDetails>();
            symmetricalDetails.Add(new
            Controllers.SymmetricalDetails("parent", "", "Server"));
            symmetricalDetails.Add(new Controllers.SymmetricalDetails("1",
            "parent", "Server"));
            symmetricalDetails.Add(new Controllers.SymmetricalDetails("2",
            "parent", "Server"));
            symmetricalDetails.Add(new Controllers.SymmetricalDetails("3",
            "1", "Server"));
        }
    }
}
```

```
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("4",
"1", "Server"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("5",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("6",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("7",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("8",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("9",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("10",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("11",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("12",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("13",
"1", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("14",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("15",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("16",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("18",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("19",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("20",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("21",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("22",
"2", "Hub"));
        symmetricalDetails.Add(new Controllers.SymmetricalDetails("23",
"2", "Hub"));
        return symmetricalDetails;
    }
}
```

Mind Map layout

A mind map is a diagram that displays the nodes as a spider diagram organizes information around a central concept. To create mind map, the [type](#) of layout should be set as **MindMap**.

Tree Orientation in layout

An [Orientation](#) of a **MindMapTreeLayout** is used to arrange the tree layout according to a specific direction. By default, the orientation is set to Horizontal. The following table outlines the various orientation types available:

Orientation Type	Description
------------------	-------------

Horizontal	Aligns the tree layout from left to right
Vertical	Aligns the tree layout from top to bottom

Note: If you want to use mind map layout in diagram, you need to inject MindMap in the diagram.

The following code example illustrates how to create an mindmap layout.

MINDMAP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = MindMapDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
};
            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class MindMapDetails
    {
        public string Id { get; set; }
        public string Label { get; set; }
        public string ParentId { get; set; }
        public string Branch { get; set; }
        public string Fill { get; set; }
        public MindMapDetails(string id, string label, string parent, string
branch, string fill)
        {
            this.Id = id;
            this.Label = label;
            this.ParentId = parent;
            this.Branch = branch;
            this.Fill = fill;
        }
        public static List<MindMapDetails> GetData()
        {
            List<MindMapDetails> mindmapDetails = new
List<Controllers.MindMapDetails>();
            mindmapDetails.Add(new Controllers.MindMapDetails("1",
"Creativity", "", "Root", "red"));
        }
    }
}
```

```

        mindmapDetails.Add(new Controllers.MindMapDetails("3",
"Brainstorming", "1", "Right", "red"));
        mindmapDetails.Add(new Controllers.MindMapDetails("4",
"Complementing", "1", "Left", "red"));
        mindmapDetails.Add(new Controllers.MindMapDetails("22",
"Sessions", "3", "subRight", "red"));
        mindmapDetails.Add(new Controllers.MindMapDetails("23",
"Complementing", "3", "subRight", "red"));
        mindmapDetails.Add(new Controllers.MindMapDetails("25", "Local",
"22", "subRight", ""));
        mindmapDetails.Add(new Controllers.MindMapDetails("26",
"Remote", "22", "subRight", ""));
        mindmapDetails.Add(new Controllers.MindMapDetails("27",
"Individual", "22", "subRight", ""));
        mindmapDetails.Add(new Controllers.MindMapDetails("28", "Teams",
"22", "subRight", ""));
        mindmapDetails.Add(new Controllers.MindMapDetails("29", "Ideas",
"23", "subRight", ""));
        mindmapDetails.Add(new Controllers.MindMapDetails("30",
"Engagement", "23", "subRight", ""));
        return mindmapDetails;
    }
}
}

```

Complex hierarchical tree

Complex hierarchical tree layout is the extended version of the hierarchical tree layout. The child has two or more parents. To create a complex hierarchical tree, the [type](#) of layout should be set as `ComplexHierarchicalTree`.

COMPLEXHIERARCHICALTREE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = ComplexHierarchicalDataDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };

            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class ComplexHierarchicalDataDetails

```

```

{
    public string Name { get; set; }
    public string Color { get; set; }
    public string[] ReportingPersons { get; set; }
    public string Border { get; set; }
    public ComplexHierarchicalDataDetails(string name, string color,
string[] reportingPersons, string border)
    {
        this.Name = name;
        this.Color = color;
        this.ReportingPersons = reportingPersons;
        this.Border = border;
    }
    public static List<ComplexHierarchicalDataDetails> GetData()
    {
        List<ComplexHierarchicalDataDetails> data = new
List<ComplexHierarchicalDataDetails>();
        data.Add(new ComplexHierarchicalDataDetails("node11", "#e7704c",
null, "#c15433"));
        data.Add(new ComplexHierarchicalDataDetails("node12", "#efd46e",
new string[] { "node114" }, "#d6b123"));
        data.Add(new ComplexHierarchicalDataDetails("node13", "#58b087",
new string[] { "node12" }, "#16955e"));
        data.Add(new ComplexHierarchicalDataDetails("node14", "#58b087",
new string[] { "node12" }, "#16955e"));
        data.Add(new ComplexHierarchicalDataDetails("node15", "#58b087",
new string[] { "node12" }, "#16955e"));
        data.Add(new ComplexHierarchicalDataDetails("node16", "#14ad85",
new string[] { }, ""));
        data.Add(new ComplexHierarchicalDataDetails("node17", "#659be5",
new string[] { "node13", "node14", "node15" }, "#3a6eb5"));
        data.Add(new ComplexHierarchicalDataDetails("node18", "#14ad85",
new string[] { }, ""));
        data.Add(new ComplexHierarchicalDataDetails("node19", "#8dbe6c",
new string[] { "node16", "node17", "node18" }, "#489911"));
        data.Add(new ComplexHierarchicalDataDetails("node110",
"#8dbe6c", new string[] { "node16", "node17", "node18" }, "#489911"));
        data.Add(new ComplexHierarchicalDataDetails("node111",
"#8dbe6c", new string[] { "node16", "node17", "node18", "node116" },
"#489911"));
        data.Add(new ComplexHierarchicalDataDetails("node21", "#e7704c",
null, "#c15433"));
        return data;
    }
}
}

```

Line Distribution

Line distribution is used to arrange the connectors without overlapping in automatic layout. In some cases, the automatic layout connectors connecting to the nodes will be overlapped with one another. So user can decide whether the segment of each connector from a single parent node should be same point or different point. The [ConnectionPointOrigin](#) property of layout is used to enable or disable the line distribution in layout. By default, ConnectionPointOrigin will be `SamePoint`.

`cs

```
public ActionResult LineDistribution()
{
    DiagramLayout connectionPointOriginValue = new DiagramLayout() { ConnectionPointOrigin =
    ConnectionPointOrigin.DifferentPoint };
    ViewBag.connectionPointOrigin = connectionPointOriginValue;
    return View();
}
`cs
```

Linear Arrangement

Linear arrangement is used to linearly arrange the child nodes in layout, which means the parent node is placed in the center corresponding to its children. When line distribution is enabled, linear arrangement is also activated by default. The [Arrangement](#) property of layout is used to enable or disable the linear arrangement in layout. By default, Arrangement will be **Nonlinear**.

Note: Linear arrangement is applicable only for complex hierarchical tree layout.

```
`cs
public ActionResult LinearArrangement()
{
    DiagramLayout arrangementValue = new DiagramLayout() { Arrangement = ChildArrangement.Linear};
    ViewBag.arrangement = arrangementValue;
    return View();
}
`cs
```

Prevent connectors overlay

The below constraints prevents the connector segments overlapping nodes with a complex hierarchical layout.

```
`cs
DiagramLayout arrangementValue = new DiagramLayout() { Arrangement = ChildArrangement.Linear};
`cs
```

Customize layout

Orientation, spacings, and position of the layout can be customized with a set of properties.

To explore layout properties, refer to [Layout Properties](#).

Layout bounds

Diagram provides support to align the layout within any custom rectangular area. For more information about bounds, refer to [bounds](#).

Layout alignment

The layout can be aligned anywhere over the layout bounds/viewport using the [horizontalAlignment](#) and [verticalAlignment](#) properties of the layout.

LAYOUTALIGN.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };

            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class HierarchicalDetails
    {
        public string Name { get; set; }
        public string Category { get; set; }
        public string FillColor { get; set; }
        public HierarchicalDetails(string name, string category, string
fillcolor)
        {
            this.Name = name;
            this.Category = category;
            this.FillColor = fillcolor;
        }
        public static List<HierarchicalDetails> GetData()
        {
            List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
            hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));
        }
    }
}
```



```

        hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
        return hierarchicaldetails;
    }
}

```

Layout spacing

Layout provides support to add space horizontally and vertically between the nodes. The [horizontalSpacing](#) and [verticalSpacing](#) properties of the layout allows you to set the space between the nodes in horizontally and vertically.

Layout margin

Layout provides support to add some blank space between the layout bounds/viewport and the layout. The [margin](#) property of the layout allows to set the blank space.

LAYOUTMARGIN.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
        };

            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class HierarchicalDetails
    {
        public string Name { get; set; }
        public string Category { get; set; }
        public string FillColor { get; set; }
        public HierarchicalDetails(string name, string category, string
fillcolor)
        {
            this.Name = name;
            this.Category = category;

```

```

        this.FillColor = fillcolor;
    }
    public static List<HierarchicalDetails> GetData()
    {
        List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
        hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
        return hierarchicaldetails;
    }
}

```

Layout orientation

The layout orientation can be used to arrange the layout based on the direction. There are different orientation types that are defined in the following table.

Orientation	Description
TopToBottom	Aligns the layout from top to bottom. All the roots are placed at top of diagram.
LeftToRight	Aligns the layout from left to right. All the roots are placed at left of diagram.
BottomToTop	Aligns the layout from bottom to top. All the roots are placed at bottom of the diagram.
RightToLeft	Aligns the layout from right to left. All the roots are placed at right of the diagram.

Diagram provides support to customize the [orientation](#) of layout. You can set the desired orientation using layout.orientation.

Note: In the diagram the default orientation is TopToBottom.

ORIENTATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            ViewBag.GetLayoutInfo = "getLayoutInfo";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
};
            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class HierarchicalDetails
    {
        public string Name { get; set; }
        public string Category { get; set; }
        public string FillColor { get; set; }
        public HierarchicalDetails(string name, string category, string
fillcolor)
        {
            this.Name = name;
            this.Category = category;
            this.FillColor = fillcolor;
        }
        public static List<HierarchicalDetails> GetData()
        {
            List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
            hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));

```

```

        hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
        return hierarchicaldetails;
    }
}

```

Fixed node

Layout provides support to arrange the nodes with reference to the position of a fixed node and set it to the [fixedNode](#) of the layout property. This is helpful when you try to expand or collapse a node. It might be expected that the position of the double-clicked node should not be changed.

FIXEDNODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            return View();
        }
    }
    public class HierarchicalDetails
    {
        public string Name { get; set; }
        public string Category { get; set; }
        public string FillColor { get; set; }
        public HierarchicalDetails(string name, string category, string
fillcolor)
        {
            this.Name = name;
            this.Category = category;
            this.FillColor = fillcolor;
        }
        public static List<HierarchicalDetails> GetData()
        {
            List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
            hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));

```

```

        hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
        return hierarchicaldetails;
    }
}
}

```

Expand and collapse

Diagram allows to expand/collapse the subtrees of a layout. The node's `isExpanded` property allows you to expand or collapse its children.

EXPANDCOLLAPSE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            DiagramMargin margin = new DiagramMargin() { Left = 10, Top = 50
};
            ViewBag.marginValue = margin;
            return View();
        }
    }
    public class HierarchicalDetails

```

```

{
    public string Name { get; set; }
    public string Category { get; set; }
    public string FillColor { get; set; }
    public HierarchicalDetails(string name, string category, string
fillcolor)
    {
        this.Name = name;
        this.Category = category;
        this.FillColor = fillcolor;
    }
    public static List<HierarchicalDetails> GetData()
    {
        List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
        hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
        hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
        return hierarchicaldetails;
    }
}
}

```

```
`javascript
```

```

function getNodeDefaults(obj, diagram) {
    obj.expandIcon = {
        height: 15,
        width: 15,
        shape: "Plus",
        fill: 'lightgray',

```

```
offset: {  
  x: .5,  
  y: .85  
}  
}  
obj.collapseIcon.offset = {  
  x: .5,  
  y: .85  
}  
obj.collapseIcon.height = 15;  
obj.collapseIcon.width = 15;  
obj.collapseIcon.shape = "Minus";  
obj.height = 50;  
obj.borderColor = 'white';  
obj.backgroundColor = '#6BA5D7';  
obj.borderWidth = 1;  
obj.style = {  
  fill: 'transparent',  
  strokeWidth: 2  
};  
}  
`
```

In the previous example, while expanding or collapsing a node, it is set as fixed node in order to prevent it from repositioning.

[Refresh layout](#)

Diagram allows to refresh the layout at runtime. To refresh the layout, refer to [Refresh layout](#).

[setNodeTemplate](#)

The `setNodeTemplate` function is provided for the purpose of customizing nodes. It will be called for each node on node initialization. In this function, the node style and its properties can be customized and can bind the custom JSON with node.

SETNODETEMPLATE.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using Syncfusion.EJ2.Diagrams;
```

```

namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            ViewBag.nodes = HierarchicalDetails.GetData();
            ViewBag.getNodeDefaults = "getNodeDefaults";
            ViewBag.getConnectorDefaults = "getConnectorDefaults";
            ViewBag.setNodeTemplate = "setNodeTemplate";
            return View();
        }
    }

    public class HierarchicalDetails
    {
        public string Name { get; set; }
        public string Category { get; set; }
        public string FillColor { get; set; }
        public HierarchicalDetails(string name, string category, string
fillcolor)
        {
            this.Name = name;
            this.Category = category;
            this.FillColor = fillcolor;
        }
        public static List<HierarchicalDetails> GetData()
        {
            List<HierarchicalDetails> hierarchicaldetails = new
List<HierarchicalDetails>();
            hierarchicaldetails.Add(new HierarchicalDetails("Diagram", "",
"#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Layout",
"Diagram", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Tree Layout",
"Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Organizational
Chart", "Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Hierarchical
Tree", "Tree Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Radial Tree",
"Tree Layout", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Mind Map",
"Hierarchical Tree", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Family Tree",
"Hierarchical Tree", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Management",
"Organizational Chart", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Human
Resources", "Management", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("University",
"Management", "#916DAF"));
            hierarchicaldetails.Add(new HierarchicalDetails("Business",
"Management", "#916DAF"));
            return hierarchicaldetails;
        }
    }
}

```



```
}
```

```
`javascript
```

```
function setNodeTemplate(obj, diagram) {
```

```
  obj.style.borderColor = obj.data.color;
```

```
}
```

```
,
```

Accessibility

Diagram provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA Accessibility supports are achieved through the attributes like `aria-label`. It helps to provide information about elements in a document for assistive technology.

The accessibility compliance for the diagram component is outlined below.

Accessibility Criteria	Compatibility
-----	-----

WCAG 2.2 Support	
Section 508 Support	
Screen Reader Support	
Right-To-Left Support	
Color Contrast	
Mobile Device Support	
Keyboard Navigation Support	
Accessibility Checker Validation	
Axe-core Accessibility Validation	
<style>	
.post .post-content img {	
display: inline-block;	
margin: 0.5em 0;	

```
}  
</style>  
  
<div> - All  
features of the component meet the requirement.</div>  
  
<div> - Some features of the component do not meet the requirement.</div>  
  
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Diagram component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Diagram component:

| Attributes | Purpose |

| --- | --- |

| **aria-label** | Provides an accessible name for the Diagram Objects. |

Aria-label

Attribute provides the text label with some default description for below elements in diagram.

<!-- markdownlint-disable MD033 -->

Element	Default description
ResizeNorthWest	Thumb to resize the selected object on the top-left corner.
ResizeNorthEast	Thumb to resize the selected object on the top-right side direction.
ResizeSouthWest	Thumb to resize the selected object on the bottom-left side direction.
ResizeSouthEast	Thumb to resize the selected object on the bottom-right side direction.
ResizeNorth	Thumb to resize the selected object on the top side direction.
ResizeSouth	Thumb to resize the selected object on the bottom side direction.
ResizeWest	Thumb to resize the selected object on the left side direction.
ResizeEast	Thumb to resize the selected object on the right side direction.
ConnectorSourceThumb	Thumb to move the source point of the connector.
ConnectorTargetThumb	Thumb to move the target point of the connector.
RotateThumb	Thumb to rotate the selected object.

Mobile device support

Syncfusion Diagram component are more user-friendly and accessible to individuals using mobile devices, including those with disabilities. These are designed to be responsive, adaptable to various screen sizes and orientations, and touch-friendly.

Screen Reader Support

The Diagram component supports and its information was dictated properly by the screen readers based on the ARIA attributes and content.

Keyboard navigation support

Syncfusion Diagram component support keyboard navigation, allowing users who rely on alternate methods to effortlessly navigate and interact with the component.

Keyboard interaction

The Diagram component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Diagram component.

| **Command** | **Action** |

| --- | --- |

| Ctrl + A | Select All |

| Ctrl + X | Cut |

| Ctrl + C | Copy |

| Ctrl + V | Paste |

| Ctrl + Z | Undo |

| Ctrl + Y | Redo |

| Delete | Delete |

| Up Arrow | Move selected object to up |

| Down Arrow | Move selected object to down |

| Left Arrow | Move selected object to left |

| Right Arrow | Move selected object to right |

| Enter | Start Annotation Edit |

| Escape | End Annotation Edit |

Ensuring accessibility

The Diagram component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Diagram component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Diagram component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Commands

<!-- markdownlint-disable MD010 -->

There are several commands available in the diagram as follows.

- Alignment commands
- Spacing commands
- Sizing commands
- Clipboard commands
- Grouping commands
- Z-order commands
- Zoom commands
- Nudge commands
- FitToPage commands
- Undo/Redo commands

Align

Alignment commands enables to align the selected or defined objects such as nodes and connectors with respect to the selection boundary. Refer to [align](#) commands which shows how to use align methods in the diagram.

Parameters	Description	
Alignment Options	Defines the specific direction, with respect to which the objects to be aligned. The accepted values of the argument "alignment options" are as follows.	
	Left	Aligns all the selected objects at the left of the selection boundary.
	Right	Aligns all the selected objects at the right of the selection boundary.
	Center	Aligns all the selected objects at the center of the selection boundary.
	Top	Aligns all the selected

		objects at the top of the selection boundary.
	Bottom	Aligns all the selected objects at the bottom of the selection boundary.
	Middle	Aligns all the selected objects at the middle of the selection boundary.

</tr>

<tr>

<td>Objects</td>

<td><p align="left">Defines the objects to be aligned. This is an optional parameter. By default, all the nodes and connectors in the selected region of the diagram gets aligned.</p></td>

</tr>

<tr>

<td>

[Alignment Mode](#)</td>

Defines the specific mode, with respect to which the objects to be aligned. This is an optional parameter. The default alignment mode is `Object`.

The accepted values of the argument "alignment mode" are as follows.

Object Aligns the objects based on the first object in the selected list.

Selector Aligns the objects based on the selection boundary.

</td>

</tr>

</table>

ALIGN.CS

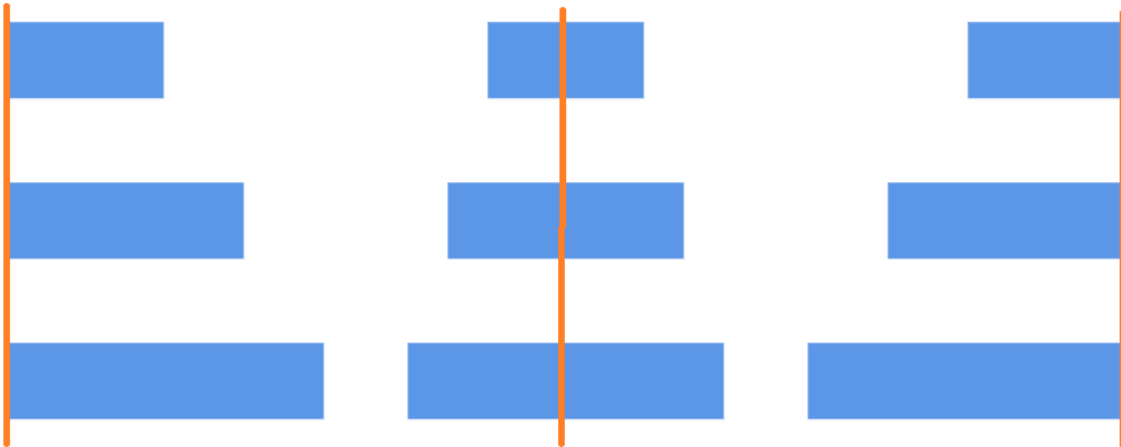
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Annotations =
Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240, Annotations =
Node3
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
}
```

```
`javascript
var diagram = document.getElementById("container").ej2_instances[0];
var selArray = [];
selArray.push(diagram.nodes[0], diagram.nodes[1], diagram.nodes[2]);
//Selects the nodes
diagram.select(selArray);
//Sets direction as left
diagram.align('Left', diagram.selectedItems.nodes, 'Selector');
diagram.dataBind();
`
```



Distribute

The [Distribute](#) commands enable to place the selected objects on the page at equal intervals from each other. The selected objects are equally spaced within the selection boundary.

The factor to distribute the shapes [DistributeOptions](#) are listed as follows:

- RightToLeft: Distributes the objects based on the distance between the right and left sides of the adjacent objects.
- Left: Distributes the objects based on the distance between the left sides of the adjacent objects.
- Right: Distributes the objects based on the distance between the right sides of the adjacent objects.
- Center: Distributes the objects based on the distance between the center of the adjacent objects.
- BottomToTop: Distributes the objects based on the distance between the bottom and top sides of the adjacent objects.

- Top: Distributes the objects based on the distance between the top sides of the adjacent objects.
- Bottom: Distributes the objects based on the distance between the bottom sides of the adjacent objects.
- Middle: Distributes the objects based on the distance between the vertical center of the adjacent objects.

DISTRIBUTE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 240, OffsetY = 100, Annotations =
Node2
            });
            nodes.Add(new Node() {
```



```

Node3
    Id = "node3", OffsetX = 170, OffsetY = 150, Annotations =
    });
    ViewBag.nodes = nodes;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

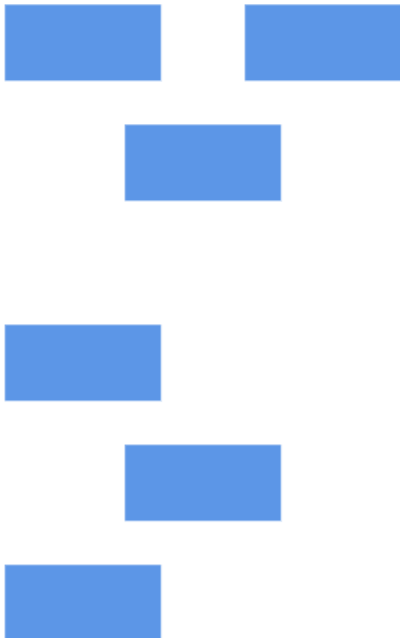
```

```
`javascript
```

```

var diagram = document.getElementById("container").ej2_instances[0];
var selArray = [];
selArray.push(diagram.nodes[0], diagram.nodes[1], diagram.nodes[2]);
//Selects the nodes
diagram.select(selArray);
//Distributes space between the nodes
diagram.distribute('RightToLeft', diagram.selectedItems.nodes);
`

```



Sizing

Sizing [sameSize](#) commands enable to equally size the selected nodes with respect to the first selected object.

[SizingOptions](#) are as follows:

- Width: Scales the width of the selected objects.
- Height: Scales the height of the selected objects.
- Size: Scales the selected objects both vertically and horizontally.

SIZE.CS

```

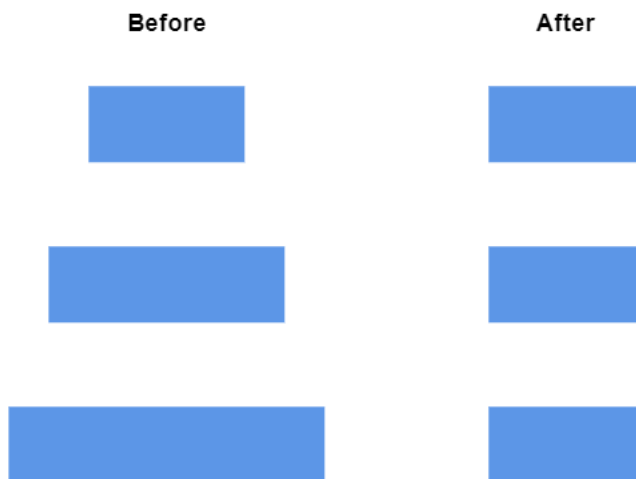
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill= "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240,Width = 120,
                Height = 80, Annotations = Node3
            });
        }
    }
}

```

```
        ViewBag.nodes = nodes;
        return View();
    }
}
public class Node: DiagramNode {
    public string text;
}
}
```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
var selArray = [];
selArray.push(diagram.nodes[0], diagram.nodes[1], diagram.nodes[2]);
//Selects the nodes
diagram.select(selArray);
//Resizes the selected nodes with the same width
diagram.sameSize('Width', diagram.selectedItems.nodes);
`
```



Clipboard

Clipboard commands are used to cut, copy, or paste the selected elements.

- Cuts the selected elements from the diagram to the diagram's clipboard, [cut](#).
- Copies the selected elements from the diagram to the diagram's clipboard, [copy](#).
- Pastes the diagram's clipboard data (nodes/connectors) into the diagram, [paste](#).

PASTE.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
                Height = 80, Annotations = Node3
            });
            List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
            Connectors.Add(new DiagramConnector() {
                Id = "connector1", SourceID = "node1", TargetID = "node2"
            });
            ViewBag.nodes = nodes;
            ViewBag.connectors = connectors;
            return View();
        }
    }
}

```

```

    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```

var diagram = document.getElementById("container").ej2_instances[0];
diagram.select([diagram.nodes[0], diagram.nodes[1], diagram.connectors[0]]);
//copies the selected nodes
diagram.copy();
//pastes the copied objects
diagram.paste(diagram.copy() as(NodeModel | ConnectorModel)[]);
`

```

Grouping

Grouping commands are used to group or ungroup the selected elements on the diagram.

[Group](#) the selected nodes and connectors in the diagram.

[Ungroup](#) the selected nodes and connectors in the diagram.

GROUP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });

```

```

    });
    nodes.Add(new Node() {
        Id = "node1",
        Width = 90,
        Height = 50,
        Style = new NodeStyleNodes() {
            fill = "#6BA5D7",
            strokeColor = "White"
        },
        text = "node1",
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1,
    });
    nodes.Add(new Node() {
        Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
        Height = 90, Annotations = Node2
    });
    nodes.Add(new Node() {
        Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
        Height = 80, Annotations = Node3
    });
    List < DiagramConnector > Connectors = new List <
DiagramConnector > ();
    Connectors.Add(new DiagramConnector() {
        Id = "connector1", SourceID = "node1", TargetID = "node2"
    });
    ViewBag.nodes = nodes;
    ViewBag.connectors = connectors;
    return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
diagram.appendTo('#element');
```

```
//Selects the diagram
```

```
diagram.selectAll();
```

```
//Groups the selected elements.
```

```
diagram.group();
```

```
,
```

[Z-Order command](#)

Z-Order commands enables to visually arrange the selected objects such as nodes and connectors on the page.

bringToFront command

The [bringToFront](#) command visually brings the selected element to front over all the other overlapped elements. The following code illustrates how to execute the `bringToFront` command.

ORDER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240,Width = 120,
                Height = 80, Annotations = Node3
            });
            ViewBag.nodes = nodes;
        }
    }
}
```

```

        return View();
    }
}
public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
let selArray: (NodeModel)[] = [];
```

```
diagram.appendTo('#element');
```

```
selArray.push(diagram.nodes[2]);
```

```
//Selects the nodes
```

```
diagram.select(selArray);
```

```
//Brings to front
```

```
diagram.bringToFront();
```

```
,
```

[sendToBack](#) command

The [sendToBack](#) command visually moves the selected element behind all the other overlapped elements. The following code illustrates how to execute the `sendToBack` command.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });

```



```

        List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
        Node3.Add(new DiagramNodeAnnotation() {
            Content = "Node3"
        });
        nodes.Add(new Node() {
            Id = "node1",
            Width = 90,
            Height = 50,
            Style = new NodeStyleNodes() {
                fill = "#6BA5D7",
                strokeColor = "White"
            },
            text = "node1",
            OffsetX = 100,
            OffsetY = 100,
            Annotations = Node1,
        });
        nodes.Add(new Node() {
            Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
            Height = 90, Annotations = Node2
        });
        nodes.Add(new Node() {
            Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
            Height = 80, Annotations = Node3
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
let selArray: (NodeModel)[] = [];
```

```
diagram.appendTo('#element');
```

```
selArray.push(diagram.nodes[2]);
```

```
//Selects the nodes
```

```
diagram.select(selArray);
```

```
//Sends to back
```

```
diagram.sendToBack();
```

```
,
```

moveForward command

The [moveForward](#) command visually moves the selected element over the nearest overlapping element. The following code illustrates how to execute the `moveForward` command.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240,Width = 120,
                Height = 80, Annotations = Node3
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {

```

```

        public string text;
    }
}

```

```
`javascript
```

```

var diagram = document.getElementById("container").ej2_instances[0];
let selArray: (NodeModel)[] = [];
diagram.appendTo('#element');
selArray.push(diagram.nodes[1]);
//Selects the nodes
diagram.select(selArray);
//Moves forward
diagram.moveForward();
`

```

sendBackward command

The [sendBackward](#) command visually moves the selected element behind the underlying element. The following code illustrates how to execute the [sendBackward](#) command.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
        }
    }
}

```

```

        nodes.Add(new Node() {
            Id = "node1",
            Width = 90,
            Height = 50,
            Style = new NodeStyleNodes() {
                fill = "#6BA5D7",
                strokeColor = "White"
            },
            text = "node1",
            OffsetX = 100,
            OffsetY = 100,
            Annotations = Node1,
        });
        nodes.Add(new Node() {
            Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
            Height = 90, Annotations = Node2
        });
        nodes.Add(new Node() {
            Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
            Height = 80, Annotations = Node3
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

public class Node: DiagramNode {
    public string text;
}
}

```

`javascript

```

var diagram = document.getElementById("container").ej2_instances[0];

let selArray: (NodeModel)[] = [];

diagram.appendTo('#element');

selArray.push(diagram.nodes[1]);

diagram.select(selArray);

//Sends backward

diagram.sendBackward();
`

```

Zoom

The [zoom](#) command is used to zoom-in and zoom-out the diagram view.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;

```

```

using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
                Height = 80, Annotations = Node3
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
// Sets the zoomFactor
//Defines the focusPoint to zoom the Diagram with respect to any point
//When you do not set focus point, zooming is performed with reference to the center of current
Diagram view.
diagram.zoom(1.2, {
x: 100,
y: 100
});
\
```

Nudge command

The [nudge](#) commands move the selected elements towards up, down, left, or right by 1 pixel.

[NudgeDirection](#) nudge command moves the selected elements towards the specified direction by 1 pixel, by default.

The accepted values of the argument "direction" are as follows:

- Up: Moves the selected elements towards up by the specified delta value.
- Down: Moves the selected elements towards down by the specified delta value.
- Left: Moves the selected elements towards left by the specified delta value.
- Right: Moves the selected elements towards right by the specified delta value.

ORDER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
```

```

Node3.Add(new DiagramNodeAnnotation() {
    Content = "Node3"
});
nodes.Add(new Node() {
    Id = "node1",
    Width = 90,
    Height = 50,
    Style = new NodeStyleNodes() {
        fill = "#6BA5D7",
        strokeColor = "White"
    },
    text = "node1",
    OffsetX = 100,
    OffsetY = 100,
    Annotations = Node1,
});
nodes.Add(new Node() {
    Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
    Height = 90, Annotations = Node2
});
nodes.Add(new Node() {
    Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
    Height = 80, Annotations = Node3
});
ViewBag.nodes = nodes;
return View();
}
}

public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
//Nudges to right
```

```
diagram.nudge('Right');
```

```
,
```

Nudge by using arrow keys

The corresponding arrow keys are used to move the selected elements towards up, down, left, or right direction by 1 pixel.



Nudge commands are particularly useful for accurate placement of elements.

BringIntoView

The [bringIntoView](#) command brings the specified rectangular region into the viewport of the diagram.

The following code illustrates how to execute the `bringIntoView` command.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240,Width = 120,
                Height = 80, Annotations = Node3
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {

```



```

        public string text;
    }
}

```

```
`javascript
```

```
var diagram = document.getElementById("container").ej2_instances[0];
```

```
//Brings the specified rectangular region of the Diagram content to the viewport of the page.
```

```
var bound = new Rect(200, 400, 500, 400);
```

```
diagram.bringIntoView(bound);
```

```
,
```

BringToCenter

The [bringToCenter](#) command brings the specified rectangular region of the diagram content to the center of the viewport.

The following code illustrates how to execute the `bringToCenter` command.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {

```

```

        fill = "#6BA5D7",
        strokeColor = "White"
    },
    text = "node1",
    OffsetX = 100,
    OffsetY = 100,
    Annotations = Node1,
});
nodes.Add(new Node() {
    Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
    Height = 90, Annotations = Node2
});
nodes.Add(new Node() {
    Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
    Height = 80, Annotations = Node3
});
ViewBag.nodes = nodes;
return View();
}
}
public class Node: DiagramNode {
    public string text;
}
}

```

```
`javascript
```

```

var diagram = document.getElementById("container").ej2_instances[0];
//Brings the specified rectangular region of the Diagram content to the center of the viewport.
let bound: Rect = new Rect(200, 400, 500, 400);
diagram.bringToCenter(bound);
`

```

FitToPage command

The [fitToPage](#) command helps to fit the diagram content into the view with respect to either width, height, or at the whole.

The [mode](#) parameter defines whether the diagram has to be horizontally/vertically fits into the viewport with respect to width, height, or entire bounds of the diagram.

The [region](#) parameter defines the region that has to be drawn as an image.

The [margin](#) parameter defines the region/bounds of the diagram content that is to be fit into the view.

The [canZoomIn](#) parameter enables/disables zooming to fit the smaller content into a larger viewport.

The [customBounds](#) parameter the custom region that has to be fit into the viewport.

The following code illustrates how to execute `FitToPage` command.

ORDER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 90,
                Height = 50,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Width = 60,
                Height = 90, Annotations = Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240, Width = 120,
                Height = 80, Annotations = Node3
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
var diagram = document.getElementById("container").ej2_instances[0];
//fit the diagram to the page with respect to mode and region
diagram.fitToPage({
mode: 'Page',
region: 'Content',
margin: {
bottom: 50
},
canZoomIn: false
});
`
```

Command manager

Diagram provides support to map or bind command execution with desired combination of key gestures. Diagram provides some built-in commands. [CommandManager](#) provides support to define custom commands. The custom commands are executed, when the specified key gesture is recognized.

Custom command

To define a custom command, specify the following properties:

- [execute](#): A method to be executed.
- [canExecute](#): A method to define whether the command can be executed at the moment.
- [gesture](#): A combination of [keys](#) and [KeyModifiers](#).
- [parameter](#): Defines any additional parameters that are required at runtime.
- [name](#): Defines the name of the command.

To explore the properties of custom commands, refer to [Commands](#).

EXISTINGCOMMAND.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController: Controller
    {
        // GET: Node
        public ActionResult Node()
        {
```

```

        // Sets the Annotation for the Node
        List<DiagramNode> Nodes = new List<DiagramNode>();
        List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
        Node1.Add(new DiagramNodeAnnotation()
        {
            //Sets the offset for the content
            Content = "Node1",
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent"
            },
        });

        List<DiagramCommand> command = new List<DiagramCommand>();
        DiagramCommand command1 = new DiagramCommand()
        {
            CanExecute = "canExecute",
            Name = "customCopy",
            Parameter = "Node",
            Gesture = new DiagramKeyGesture() { Key = Keys.G,
KeyModifiers = KeyModifiers.Shift | KeyModifiers.Alt },
            Execute = "execute"
        };
        command.Add(command1);

        Nodes.Add(new DefaultNode()
        {
            Id = "Node1",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width = 100,
            // add the Annotation for the Node
            Annotations = Node1,
        });

        ViewBag.nodes = Nodes;
        ViewBag.Command = command;
        return View();
    }
}

```

```

`javascript

```

```

function canExecute() {
var diagram = document.getElementById("diagram").ej2_instances[0];
//Defines that the clone command can be executed, if and only if the selection list is not empty.
if (diagram.selectedItems.nodes.length > 0 || diagram.selectedItems.connectors.length > 0) {
return true;
}
}

```

```

}
return false;
}
//Command handler
function execute() {
//Logic to clone the selected element
var diagram = document.getElementById("diagram").ej2_instances[0];
diagram.copy();
diagram.paste();
diagram.dataBind();
}
`

```

Modify the existing command

When any one of the default commands is not desired, they can be disabled. To change the functionality of a specific command, the command can be completely modified.

CSHTML

```

<div class="col-lg-8 control-section" style="float:left;padding:0px">
    <div class="content-wrapper">
        @(Html.EJS().Diagram("diagram")
            .Width("100%")
            .Height("500px")
            .Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG)
            .CommandManager(g=> g.Commands(ViewBag.Command))
            .Nodes(ViewBag.nodes).Render())
    </div>
</div>

```

NUDGE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController: Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node

```

```

        List<DiagramNode> Nodes = new List<DiagramNode>();
        List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
        Node1.Add(new DiagramNodeAnnotation()
        {
            //Sets the offset for the content
            Content = "Node1",
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent"
            },
        });

        List<DiagramCommand> command = new List<DiagramCommand>();
        DiagramCommand command1 = new DiagramCommand()
        {
            Name = "delete"
            Parameter = "node",
            Execute = "canExecute"
        };
        command.Add(command1);

        Nodes.Add(new DefaultNode()
        {
            Id = "Node1",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width = 100,
            // add the Annotation for the Node
            Annotations = Node1,
        });

        ViewBag.nodes = Nodes;
        ViewBag.Command = command;
        return View();
    }
}

```

`javascript

```

function canExecute() {
return false;
}
`

```

See Also

- [How to create the custom context menu items](#)

History List in Diagram

Diagram tracks the history of actions that are performed after initializing the diagram and provides support to reverse and restore those changes.

Undo and redo

Diagram provides built-in support to track the changes that are made through interaction and through public APIs. The changes can be reverted or restored either through shortcut keys or through commands.

Undo/redo through shortcut keys

Undo/redo commands can be executed through shortcut keys. Shortcut key for undo is Ctrl+z and shortcut key for redo is Ctrl+y.

Undo/redo through public APIs

The client-side methods [undo](#) and [redo](#) helps to revert/restore the changes.

```
` javascript
var diagram = document.getElementById("diagram").ej2_instances[0];
// Reverts the last action performed
diagram.undo();
// Restores the last undone action
diagram.redo();
`
```

When a change in the diagram is reverted or restored (undo/redo), the historyChange event gets triggered.

Group multiple changes

History list allows to revert or restore multiple changes through a single undo/redo command. For example, revert/restore the fill color change of multiple elements at a time.

The client-side method [startGroupAction](#) is used to notify the diagram to start grouping the changes. The client-side method [endGroupAction](#) is used to notify to stop grouping the changes.

CSHTML

```
<div class="col-lg-12 control-section">
  <div class="content-wrapper">
    @(Html.EJS().Diagram("container")
      .Width("100%")
      .Height("500px")
      .Nodes(ViewBag.nodes).Render())
  </div>
</div>
```

UNDOREDO.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```



```

using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "Node2"
            });
            List < DiagramNodeAnnotation > Node3 = new List <
DiagramNodeAnnotation > ();
            Node3.Add(new DiagramNodeAnnotation() {
                Content = "Node3"
            });
            nodes.Add(new Node() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    fill = "#6BA5D7",
                    strokeColor = "White"
                },
                text = "node1",
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1,
            });
            nodes.Add(new Node() {
                Id = "node2", OffsetX = 100, OffsetY = 170, Annotations =
Node2
            });
            nodes.Add(new Node() {
                Id = "node3", OffsetX = 100, OffsetY = 240, Annotations =
Node3
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
    public class Node: DiagramNode {
        public string text;
    }
}

```

```
`javascript
```

```
var diagram = document.getElementById('container').ej2_instances[0];
diagram.startGroupAction();
//Makes the changes
var color = ['black', 'red', 'green', 'yellow']
for (var i = 0; i < color.length; i++) {
// Updates the fillColor for all the child elements.
diagram.nodes[0].style.fill = color[i];
diagram.dataBind();
}
//Ends grouping the changes
diagram.endGroupAction();
`
```

Track custom changes

Diagram provides options to track the changes that are made to custom properties. For example, in case of an employee relationship diagram, track the changes in the employee information. The historyList of the diagram enables to track such changes.

Before changing the employee information, save the existing information to historyList by using the client-side method push of historyList. The historyList canLog method can be used which takes a history entry as argument and returns whether the specific entry can be added or not.

```
`javascript
var diagram = document.getElementById('container').ej2_instances[0];
//Creates a custom entry
var entry = {
undoObject: diagram.nodes[0];
};
// adds that to history list
diagram.historyList.push(entry);
diagram.dataBind();
`
```

canLog

canLog in the history list, which takes a history entry as argument and returns whether the specific entry can be added or not.

```
`javascript
var diagram = document.getElementById('container').ej2_instances[0];
diagram.historyList.canLog = function(entry) {
```

```
entry.cancel = true;
return entry;
}
`
```

Track undo/redo actions

The `historyList.undoStack` property is used to get the collection of undo actions which should be performed in the diagram. The `undoStack/redoStack` is the read-only property.

```
`javascript
var diagram = document.getElementById('container').ej2_instances[0];
//get the collection of undoStack objects
let undoStack = diagram.historyList.undoStack;
//get the collection of redoStack objects
let redoStack = diagram.historyList.redoStack;
`
```

History change event

The [historyChange](#) event triggers, whenever the interaction of the node and connector is take place.

```
`javascript
var diagram = document.getElementById('container').ej2_instances[0];
// history change event
diagram.historyChange = (arg) => {
//causes of history change
let cause: string = arg.cause;
}
`
```

Retain Selection

You can retain a selection at undo/redo operation by using the client-side API Method called `updateSelection`. Using this method, you can select diagram objects.

```
`typescript
let diagramInstance: DiagramComponent;
ReactDOM.render( < DiagramComponent id = "diagram" ref={diagram => diagramInstance = diagram}
width = {
'100%'
}
height = {
'600px'
```

```

}
nodes = {
  nodes
}
/>, document.getElementById("diagram"));
// history change event
diagramInstance.updateSelection: (object: NodeModel, diagram: Diagram) => {
  let selArr = [];
  selArr.push(object)
  diagram.select(selArr);
},
,

```

Virtualization in Diagram

Virtualization in Diagram

Virtualization is the process of loading the diagramming objects available in the visible area of the Diagram control, that is, only the diagramming objects that lie within the ViewPort of the Scroll Viewer are loaded (remaining objects are loaded only when they come into view).

This feature gives an optimized performance while loading and dragging items to the Diagram that consists of many Nodes and Connectors.

```

`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
//Enable virtualization in diagram
diagram.constraints = DiagramConstraints.Default | DiagramConstraints.Virtualization,
diagram.dataBind();
,

```

Serialization in Diagram Control

Serialization is the process of saving and loading for state persistence of the diagram.

Save

The diagram is serialized as string while saving. The client-side method, [saveDiagram](#) helps to serialize the diagram as a string.

```

`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
var saveData;

```

```
//returns serialized string of the Diagram
```

```
saveData = diagram.saveDiagram();
```

```
,
```

This string can be converted to JSON data and stored for future use.

```
`javascript
```

```
//Saves the string in to local storage
```

```
localStorage.setItem('fileName', saveData);
```

```
saveData = localStorage.getItem('fileName');
```

```
,
```

Diagram can also be saved as raster or vector image files. For more information about saving the diagram as images, refer to [Print and Export](#).

Load

Diagram is loaded from the serialized string data by client-side method, [loadDiagram](#).

```
`javascript
```

```
var diagramElement = document.getElementById('element');
```

```
var diagram = diagramElement.ej2_instances[0];
```

```
//Loads the Diagram from saved json data
```

```
diagram.loadDiagram(saveData);
```

```
,
```

Note: Before loading a new diagram, existing diagram is cleared.

Prevent Default Values

The [preventDefaults](#) property of `serializationSettings` is used to simplifying the saved JSON object without adding the default properties that are presented in the diagram.

```
`typescript
```

```
var diagram: Diagram = new Diagram({
```

```
  serializationSettings: { preventDefaults: true },
```

```
});
```

```
,
```

Exporting in Diagram

Diagram provides support to export its content as image or svg files. The client-side method [exportDiagram](#) helps to export the diagram.

Note: To use Print and Export, you need to inject `PrintAndExport` in the diagram.

```
<!-- markdownlint-disable MD033 -->
```

```
`javascript
```

```
var diagram = new Diagram({
```

```
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
diagram.exportDiagram(options);
`
```

Exporting options

Diagram provides support to export the desired region of the diagram to desired formats.

File Name

[FileName](#) is the name of the file to be downloaded. By default, the file name is set to **Diagram**.

Format

[Format](#) is to specify the type or format of the exported file. By default, the diagram is exported as .jpg format. You can export diagram to the following formats:

- JPG
- PNG
- BMP
- SVG

```
`javascript
var diagram = new Diagram({
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.format = 'SVG';
diagram.exportDiagram(options);
`
```

Margin

[Margin](#) specifies the amount of space that has to be left around the diagram.

<!-- markdownlint-disable MD033 -->

```
`javascript
var diagram = new Diagram({
width: 1500, height: 1500
});
```

```
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
diagram.exportDiagram(options);
`
```

Mode

[Mode](#) specifies whether the diagram is to be exported as files or as data (ImageURL/SVG). The exporting options are as follows:

- Data: Exports and downloads the diagram as image.
- Download: Exports the diagram as data of formats ImageURL/SVG.

For more information about the exporting modes, refer to [Exporting Modes](#).

```
`javascript
var diagram = new Diagram({
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
diagram.exportDiagram(options);
`
```

Region

You can export any particular [region](#) of the diagram and the region is categorized as follows.

- Region that fits all nodes and connectors that are added to model.
- Region that fits all pages (single or multiple pages based on page settings).

For more information about region, refer to [Regions](#).

```
`javascript
var diagram = new Diagram({
```

```
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
options.region = 'Content';
diagram.exportDiagram(options);
`
```

Custom bounds

Diagram provides support to export any specific region of the diagram by using [bounds](#).

```
`javascript
var diagram = new Diagram({
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'region';
options.format = 'SVG';
options.region = 'CustomBounds';
options.bounds.x = 10;
options.bounds.y = 10;
options.bounds.height = 100;
options.bounds.width = 100;
diagram.exportDiagram(options);
`
```

Export diagram with stretch option

Diagram provides support to export the diagram as image for [stretch](#) option. The exported images will be clearer but larger in file size.

```
`javascript
```



```
var diagram = new Diagram({
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'region';
options.format = 'SVG';
options.region = 'Content';
options.stretch = 'Stretch';
diagram.exportDiagram(options);
`
```

Print

The client-side method [print](#) helps to print the diagram as image.

Name	Type	Description
------	------	-------------

-----	-----	-----
-------	-------	-------

region	enum	Sets the region of the diagram to be printed.
--------	------	---

bounds	object	Prints any custom region of diagram.
--------	--------	--------------------------------------

stretch	enum	Resizes the diagram content to fill its allocated space and printed.
---------	------	--

multiplePage	boolean	Prints the diagram into multiple pages.
--------------	---------	---

pageWidth	number	Sets the page width of the diagram while printing the diagram into multiple pages.
-----------	--------	--

pageHeight	number	Sets the page height of the diagram while printing the diagram into multiple pages.
------------	--------	---

pageOrientation	enum	Sets the orientation of the page.
-----------------	------	-----------------------------------

```
`javascript
```

```
var diagram = new Diagram({
width: 1500, height: 1500
});
diagram.appendTo('#element');
var options = {};
options.mode = 'Data';
options.region = 'PageSettings';
```

```
options.multiplePage = true;
options.pageHeight = 300;
options.pageWidth = 300;
diagram.print(options);
\
```

Tooltip in Diagram Control

<!-- markdownlint-disable MD010 -->

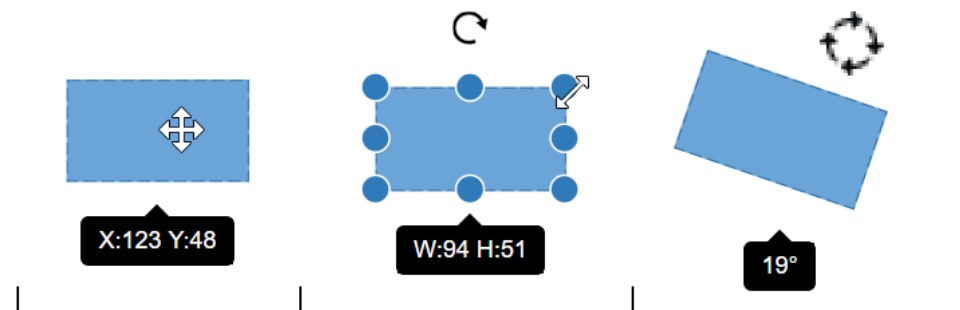
In Graphical User Interface (GUI), the tooltip is a message that is displayed when mouse hovers over an element. The diagram provides tooltip support while dragging, resizing, rotating a node, and when the mouse hovers any diagram element.

Default tooltip

By default, diagram displays a tooltip to provide the size, position, and angle related information while dragging, resizing, and rotating.

| Drag | Resize | Rotate |

|---|---|---|



Common tooltip for all nodes and connectors

The diagram provides support to show tooltip when the mouse hovers over any node or connector. To show tooltip on mouse over, the [tooltip](#) property of diagram model needs to be set with the tooltip [content](#) and [position](#).

DEFAULT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
```

```

        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "#6BA5D7",
            StrokeColor = "White"
        },
        OffsetX = 100,
        OffsetY = 100,
        // Set constraints for the node
        constraints = NodeConstraints.Default |
NodeConstraints.Tooltip,
        Tooltip = new DiagramDiagramTooltip() { Content =
"Nodes", Position = "TopLeft" }
    });
    ViewBag.nodes = nodes;
    string position = "TopLeft";
    ViewBag.position = position;
    return View();
}
}
}

```

Disable tooltip at runtime

The tooltip on mouse over can be disabled by assigning the [tooltip](#) property as `null`.

DISABLE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            string position = "TopLeft";
            ViewBag.position = position;
            return View();
        }
    }
}

```

Tooltip for a specific node/connector

The tooltip can be customized for each node and connector. Remove the **InheritTooltip** option from the [constraints](#) of that node or connector.

OBJECT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                OffsetX = 100,
                OffsetY = 100,
                Constraints = (NodeConstraints.Default &
~NodeConstraints.InheritTooltip) | NodeConstraints.Tooltip,
                Tooltip = new DiagramDiagramTooltip() { Content =
"Nodes", Position = "TopLeft", RelativeMode= TooltipRelativeMode.Object}
            });
            ViewBag.nodes = nodes;
            string position = "TopLeft";
            ViewBag.position = position;
            return View();
        }
    }
}
```

Tooltip for Ports

The tooltip feature has been implemented to support Ports, providing the ability to display information or descriptions when the mouse hovers over them.

To display tooltips on mouseover, set the desired tooltip [content](#) by utilizing the [tooltip](#) property.

Tooltips for Ports can be enabled or disabled using the [PortConstraints](#) Tooltip property.

```
`js
```

```
let ports: [{
```

```
  offset: {x: 1,y: 0.5},
```

```

tooltip: {content: 'Port Tooltip'},
//enable Port Tooltip Constraints
constraints: PortConstraints.Default | PortConstraints.ToolTip,
//disable Port Tooltip Constraints
constraints: PortConstraints.Default ~& PortConstraints.ToolTip
}}
,

```

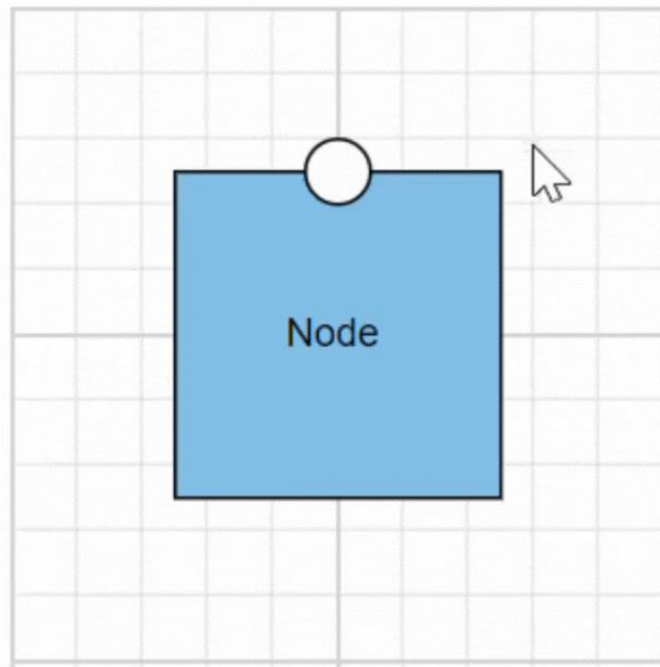
Dynamic modification of tooltip content is supported, allowing you to change the displayed tooltip content during runtime.

```

`js
{
//change tooltip content at run time
diagram.nodes[0].ports[0].tooltip.content = 'New Tooltip Content';
diagram.databind;
}
,

```

The following image illustrates how the diagram displays tooltips during an interaction with ports:



Here, the code provided below demonstrates the port tooltip Interaction.

OBJECT.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramPort > ports1 = new List < DiagramPort > ();
            ports1.Add(new CustomPort() {
                Id = "port1", Shape = PortShapes.Circle,
                Offset = new DiagramPoint() {
                    X = 0.5, Y = 0.5
                }
            });
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                OffsetX = 100,
                OffsetY = 100,
                ports= ports1
            });
            ViewBag.nodes = nodes;
            string position = "TopLeft";
            ViewBag.position = position;
            return View();
        }
    }
}

```

Tooltip template content

Any text or image can be added to the tooltip, by default. To customize the tooltip layout or to create your own visualized element on the tooltip, template can be used.

TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            string position = "TopLeft";

```

```

        ViewBag.position = position;
        ViewBag.getContent = "getContent";
        return View();
    }
}

```

```
`javascript
```

```

function getContent() {
var tooltipContent = document.createElement('div');

tooltipContent.innerHTML = '<div style="background-color: #f4f4f4; color: black; border-
width:1px;border-style: solid;border-color: #d3d3d3; border-radius: 8px;white-space: nowrap;"> <span
style="margin: 10px;"> Tooltip !!! </span> </div>';

return tooltipContent;
}
`

```

Tooltip alignments

Tooltip relative to object

The diagram provides support to show tooltip around the node or connector that is hovered by the mouse. The tooltip can be aligned by using the [position](#) property of the tooltip. The [relativeMode](#) property of the tooltip defines whether the tooltip has to be displayed around the object or at the mouse position.

OBJECT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                OffsetX = 100,
                OffsetY = 100,
                Constraints = (NodeConstraints.Default &
~NodeConstraints.InheritTooltip) | NodeConstraints.Tooltip,

```

```

        Tooltip = new DiagramDiagramTooltip() { Content =
"Nodes", Position = "TopLeft", RelativeMode= TooltipRelativeMode.Object}
    });
    ViewBag.nodes = nodes;
    string position = "TopLeft";
    ViewBag.position = position;
    return View();
}
}
}

```

Tooltip relative to mouse position

To display the tooltip at mouse position, need to set **mouse** option to the [relativeMode](#) property of the tooltip.

MOUSE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                OffsetX = 100,
                OffsetY = 100,
                Constraints = NodeConstraints.Default |
NodeConstraints.Tooltip,
                Tooltip = new DiagramDiagramTooltip() { Content =
"Nodes", Position = "TopLeft", RelativeMode= TooltipRelativeMode.Mouse}
            });
            ViewBag.nodes = nodes;
            string position = "TopLeft";
            ViewBag.position = position;
            return View();
        }
    }
}

```


Tooltip animation

To animate the tooltip, a set of specific animation effects are available, and it can be controlled by using the [animation](#) property. The animation property also allows you to set delay, duration, and various other effects of your choice.

ANIMATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "#6BA5D7",
                    StrokeColor = "White"
                },
                OffsetX = 100,
                OffsetY = 100,
                Constraints = NodeConstraints.Default |
NodeConstraints.Tooltip,
                Tooltip = new DiagramDiagramTooltip() { Content =
"Nodes", Position = "TopLeft", RelativeMode= TooltipRelativeMode.Object}
            });
            ViewBag.nodes = nodes;
            string position = "TopLeft";
            ViewBag.position = position;
            return View();
        }
    }
}
```

User Handles in Diagram

- User handles are used to add some frequently used commands around the selector. To create user handles, define and add them to the [userHandles](#) collection of the [selectedItems](#) property.
- The name property of user handle is used to define the name of the user handle and its further used to find the user handle at runtime and do any customization.

Alignment

User handles can be aligned relative to the node boundaries. It has [margin](#), [offset](#), [side](#), [horizontalAlignment](#), and [verticalAlignment](#) settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset for user handle

The [offset](#) property of [userHandles](#) is used to align the user handle based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Side

The [side](#) property of [userHandles](#) is used to align the user handle by using the [Top](#), [Bottom](#), [Left](#), and [Right](#) options.

Horizontal and vertical alignments

The [horizontalAlignment](#) property of [userHandles](#) is used to set how the user handle is horizontally aligned at the position based on the [offset](#). The [verticalAlignment](#) property is used to set how user handle is vertically aligned at the position.

Margin for the user handle

Margin is an absolute value used to add some blank space in any one of its four sides. The [userHandles](#) can be displaced with the [margin](#) property.

Appearance

The appearance of the user handle can be customized by using the [size](#), [borderColor](#), [backgroundColor](#), [visible](#), [pathData](#), and [pathColor](#) properties of the [userHandles](#).

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;

namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                Shape = new { type = "Basic", shape = "Rectangle" },
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
            List<DiagramUserHandle> Userhandle = new
List<DiagramUserHandle>();
            Userhandle.Add(new DiagramUserHandle()
            {
                Name = "clone",
                PathData = "M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3," +
```

```

        "0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
        5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z " +
        "M68.5,72.5h-30V34.4h30V72.5z",
        Visible = true,
        Offset = 0,
        Side = Side.Bottom,
        Margin = new DiagramMargin() { Left = 0, Right = 0, Top = 0,
Bottom = 0 }
    });
    ViewBag.userhandle = Userhandle;
    ViewBag.getTool = "getTool";
    return View();
}
}
}

```

```
`javascript
```

```

function getTool(action) {
    var tool;
    if (action === 'clone') {
        tool = new CloneTool(diagram.commandHandler);
    }
    return tool;
}

var extends = (this && this.extends) || (function () {
    var extendStatics = Object.setPrototypeOf ||
        / jshint proto: true /
        ({ proto: [] } instanceof Array && function (d, b) { d.proto = b; }) ||
        function (d, b) { for (var p in b) if (b.hasOwnProperty(p)) d[p] = b[p]; };
    return function (d, b) {
        extendStatics(d, b);
        function () { this.constructor = d; }
        d.prototype = b === null ? Object.create(b) : (b.prototype = b.prototype, new ());
    };
})();

var CloneTool = (function (_super) {
    extends(CloneTool, _super);
    function CloneTool() {
        return super !== null && super.apply(this, arguments) || this;
    }
}

```

```

}
CloneTool.prototype.mouseDown = function (args) {
var newObject;
if (diagram.selectedItems.nodes.length > 0) {
newObject = ej.diagrams.cloneObject(diagram.selectedItems.nodes[0]);
}
else {
newObject = ej.diagrams.cloneObject(diagram.selectedItems.connectors[0]);
}
newObject.id += ej.diagrams.randomId();
diagram.paste([newObject]);
args.source = diagram.nodes[diagram.nodes.length - 1];
args.sourceWrapper = args.source.wrapper;
_super.prototype.mouseDown.call(this, args);
this.inAction = true;
};
return CloneTool;
}(ej.diagrams.MoveTool));

```

Fixed user handles

The fixed user handles are used to add some frequently used commands around the node and connector even without selecting it.

Initialization an fixed user handles

To create the fixed user handles, define and add them to the collection of nodes and connectors property.

NODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            // A fixed user handle is created and stored in fixed user
handle collection of Node.

```

```

        List<DiagramNodeFixedUserHandle> handle = new
List<DiagramNodeFixedUserHandle>();
        handle.Add(new DiagramNodeFixedUserHandle() { Margin = new
DiagramMargin() {Right=20 },Width=20,Height=20, PathData = "M60.3,18H27.5c-
3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z" });
        nodes.Add(new DiagramNode() {
            Id = "node1",
            Width = 100,
            Height = 100,
            Style = new NodeStyleNodes() {
                Fill = "darkcyan"
            },
            OffsetX = 100,
            OffsetY = 100,
            FixedUserHandles = handle
        });
        ViewBag.nodes = nodes;
        return View();
    }
}

```

Customization

- The id property of fixed user handle is used to define the unique identification of the fixed user handle and it is further used to add custom events to the fixed user handle.
- The fixed user handle can be positioned relative to the node and connector boundaries. It has offset, padding and cornerRadius settings. It is used to position and customize the fixed user handle.
- The **Padding** is used to leave the space that is inside the fixed user handle between the icon and border.
- The corner radius allows to create fixed user handles with rounded corners. The radius of the rounded corner is set with the **cornerRadius** property.

Note: The PathData needs to be provided to render fixed user handle.

Size

Diagram allows to set size for the fixed user handles by using the **width** and **height** property. The default value of the width and height property is 10.

Style

- You can change the style of the fixed user handles with the specific properties of **borderColor**, **borderWidth**, and background color using the **handleStrokeColor**, **handleStrokeWidth**, and **fill** properties, and the icon **borderColor**, and **borderWidth** using the **iconStrokeColor** and **iconStrokeWidth**.
- The fixed user handle's **iconStrokeColor** and **iconStrokeWidth** property are used to change the stroke color and stroke width of the given **pathData**.

- The fixed user handle `handleStrokeColor` and `fill` properties are used to define the background color and border color of the userhandle and the `handleStrokeWidth` property is used to define the border width of the fixed user handle.
- The `visible` property of the fixed user handle enables or disables the visibility of fixed user handle.

FIXEDUSERHANDLE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
namespace sample1.Controllers
{
    public class NodeController : Controller
    {
        public ActionResult Node()
        {
            List<DiagramNode> Nodes = new List<DiagramNode>();
            // A fixed user handle is created and stored in fixed user
            handle collection of Node.
            List<DiagramNodeFixedUserHandle> nodehandle = new
            List<DiagramNodeFixedUserHandle>();
            nodehandle.Add(new DiagramNodeFixedUserHandle() { Margin = new
            DiagramMargin() {Right=20 },Width=20,Height=20, PathData = "M60.3,18H27.5c-
            3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
            5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
            5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z" });
            Nodes.Add(new DiagramNode()
            {
                Id = "Node1",
                OffsetY = 100,
                OffsetX = 100,
                Height = 100,
                Width =100,
                FixedUserHandles = nodehandle
            });
            ViewBag.nodes = Nodes;
            // A fixed user handle is created and stored in fixed user
            handle collection of Connector.
            List<DiagramConnectorFixedUserHandle> connectorhandle = new
            List<DiagramConnectorFixedUserHandle>();
            connectorhandle.Add(new DiagramConnectorFixedUserHandle()
            {Offset=0,Alignment= FixedUserHandleAlignment.After,Displacement= new
            DiagramPoint() {X=10, Y = 10 }, Width = 20, Height = 20, PathData =
            "M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-
            3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
            5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z" });
            List<DiagramConnector> Connectors = new
            List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id =
            "connector",FixedUserHandles = connectorhandle, SourcePoint = new
```

```
DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new DiagramPoint() { X = 300, Y = 300 } });
    ViewBag.connectors = Connectors;
    return View();
}
}
```

Note: The fixed user handle id need to be unique.

Customizing the node fixed user handle

- The node fixed user handle can be aligned relative to the node boundaries. It has **margin** and **offset** settings. It is quite useful to position the node fixed userhandle and used together and gives you more control over the node fixed user handle positioning.



Margin for the node fixed user handle

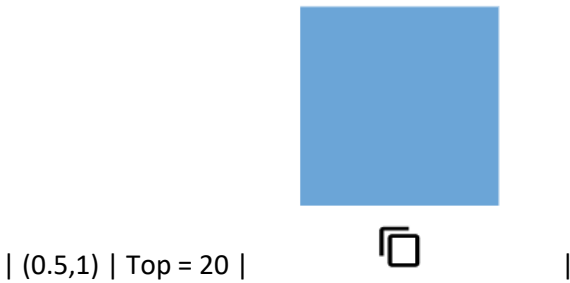
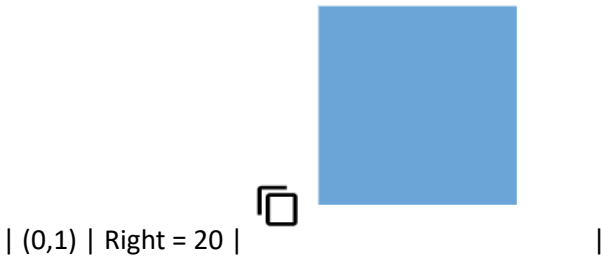
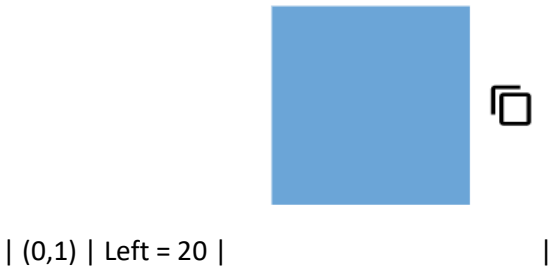
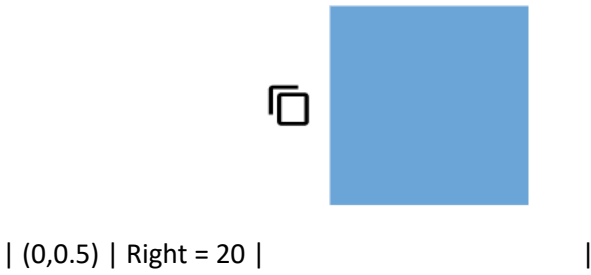
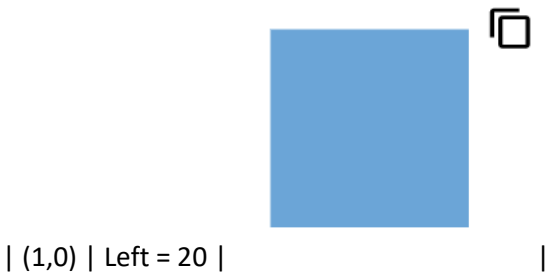
Margin is an absolute value used to add some blank space in any one of its four sides. The fixed user handle can be displaced with the **margin** property.

Offset for the node fixed user handle

The **offset** property of fixed user handle is used to align the user handle based on the **x** and **y** points. (0,0) represents the top or left corner and (1,1) represents the bottom or right corner.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Offset	Margin	Output
-----	-----	-----
(0,0)	Right = 20	
(0.5,0)	Bottom = 20	





| (1,1) | Left = 20 |

NODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            // A fixed user handle is created and stored in fixed user
handle collection of Node.
            List<DiagramNodeFixedUserHandle> handle = new
List<DiagramNodeFixedUserHandle>();
            handle.Add(new DiagramNodeFixedUserHandle() { Margin = new
DiagramMargin() {Right=20 },Width=20,Height=20, PathData = "M60.3,18H27.5c-
3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z" });
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                BorderWidth=2,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 100,
                OffsetY = 100,
                FixedUserHandles = handle
            });
            ViewBag.nodes = nodes;
            return View();
        }
    }
}
```

Customizing the connector fixed user handle

- The connector fixed user handle can be aligned relative to the connector boundaries. It has alignment, displacement and offset settings. It is useful to position the connector fixed user

handle and used together and gives you more control over the connector fixed user handle positioning.

- The **offset** and **alignment** properties of fixed user handle allows to align the connector fixed user handles to the segments.

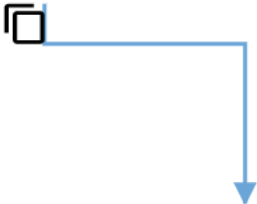
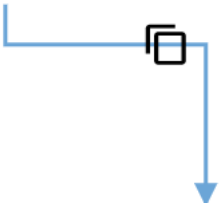

Offset for the connector fixed user handle

The **offset** property of connector fixed user handle is used to align the user handle based on fractions. 0 represents the connector source point, 1 represents the connector target point, and 0.5 represents the center point of the connector segment.

Alignment

The connector’s fixed user handle can be aligned over its segment path using the **alignment** property of fixed user handle.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Offset	Alignment	Output
-----	-----	-----
0	Before	
0.5	Center	
1	After	

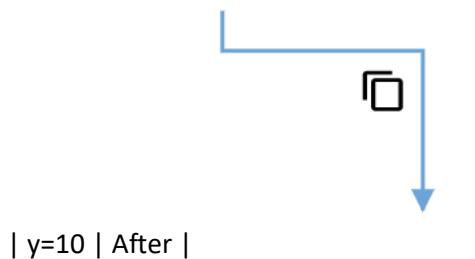
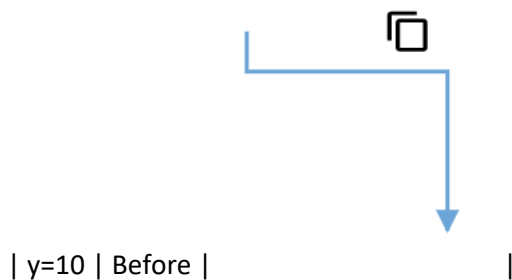
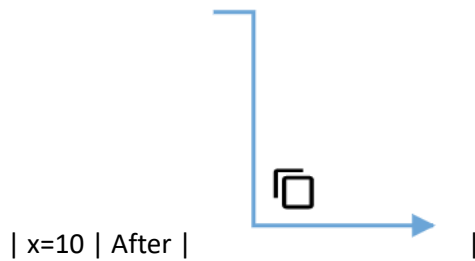
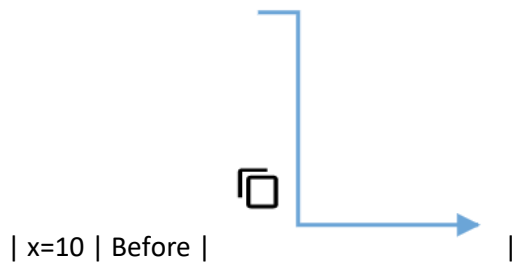
Displacement

- The **displacement** property allows you to specify the space to be left from the connector segment based on the x and y value provided.

The following table shows all the possible alignments visually shows the fixed user handle positions.

| Displacement | Alignment | Output |

| ----- | ----- | ----- |



Note: Displacement will not be done if the alignment is set to be center.

FIXEDUSERHANDLE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        public ActionResult Nodes() {
```

```

        // A fixed user handle is created and stored in fixed user
        handle collection of Connector.
        List<DiagramConnectorFixedUserHandle> handle = new
List<DiagramConnectorFixedUserHandle>();
        handle.Add(new DiagramConnectorFixedUserHandle()
{Offset=0,Alignment= FixedUserHandleAlignment.After,Displacement= new
DiagramPoint() {X=10, Y = 10 }, Width = 20, Height = 20, HandleStrokeColor =
"green", IconStrokeColor = "blue", Fill = "red", PathData =
"M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-
3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z" });
        List<DiagramConnector> Connectors = new
List<DiagramConnector>();
        Connectors.Add(new DiagramConnector() { Id =
"connector",FixedUserHandles = handle, SourcePoint = new DiagramPoint() { X
= 100, Y = 100 }, TargetPoint = new DiagramPoint() { X = 300, Y = 300 } });
        ViewBag.connectors = Connectors;
        return View();
    }
}
}

```

Tooltip support for User Handle

The diagram provides support to show tooltip when the mouse hovers over any user handle.

To show tooltip on mouse over, the [tooltip](#) property of diagram model needs to be set with the tooltip [content](#) and [position](#) as shown in the following example.

NODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                Shape = new { type = "Basic", shape = "Rectangle" },
                OffsetX = 100,
                OffsetY = 100,
            });
            ViewBag.nodes = nodes;
        }
    }
}

```

```

        List<DiagramUserHandle> Userhandle = new
List<DiagramUserHandle>();
        Userhandle.Add(new DiagramUserHandle()
        {
            Name = "clone",
            PathData = "M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3," +
            "0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z " +
            "M68.5,72.5h-30V34.4h30V72.5z",
            Visible = true,
            Offset = 0,
            Side = Side.Bottom,
            Margin = new DiagramMargin() { Left = 0, Right = 0, Top = 0,
Bottom = 0 }
        });
        ViewBag.userhandle = Userhandle;
        ViewBag.getTool = "getTool";
        return View();
    }
}

```

CSS Structure in Diagram

Customizing the connector end point handle

Use the following CSS to customize the connector end point handle.

```
`scss
```

```
.e-diagram-endpoint-handle {
```

```
fill: red;
```

```
stroke: green;
```

```
}
```

```
`
```

Customizing the connector end point handle when connected

Use the following CSS to customize the connector end point handle when connected.

```
`scss
```

```
.e-diagram-endpoint-handle.e-connected {
```

```
fill: red;
```

```
stroke: green;
```

```
}
```

```
`
```

Customizing the connector end point handle when disabled

Use the following CSS to customize the connector end point handle when disabled.

```
`scss
```

```
.e-diagram-endpoint-handle.e-disabled {  
  fill: red;  
  opacity: 1;  
  stroke: green;  
}  
`
```

Customizing the bezier connector handle

Use the following CSS to customize the bezier handle properties.

```
`scss  
.e-diagram-bezier-handle {  
  fill: red;  
  stroke: green;  
}  
`
```

Customizing the bezier connector line

Use the following CSS to customize the bezier line properties.

```
`scss  
.e-diagram-bezier-line {  
  stroke: black;  
}  
`
```

Customizing the resize handle

Use the following CSS to customize the resize handle.

```
`scss  
.e-diagram-resize-handle {  
  fill: white;  
  opacity: 1;  
  stroke: white;  
}  
`
```

Customizing the selector pivot line

Use the following CSS to customize the line between the selector and rotate handle.

```
`scss  
.e-diagram-pivot-line {
```

```
stroke: red;  
}
```

,

Customizing the selector border

Use the following CSS to customize the selector border.

```
`scss  
.e-diagram-border {  
stroke: red;  
}
```

,

Customizing the rotate handle

Use the following CSS to customize the rotate handle properties.

```
`scss  
.e-diagram-rotate-handle {  
fill: red;  
stroke: green;  
}
```

,

Customizing the symbolpalette while hovering

Use the following CSS to customize the symbolpalette while hovering.

```
`scss  
.e-symbolpalette .e-symbol-hover:hover {  
background: red;  
}
```

,

Customizing the symbolpalette when selected

Use the following CSS to customize the symbolpalette when selected.

```
`scss  
.e-symbolpalette .e-symbol-selected {  
background: white;  
}
```

,

Customizing the ruler

Use the following CSS to customize the ruler properties.

```
`scss
.e-diagram .e-ruler {
background-color: red;
font-size: 13px;
}
`
```

[Customizing the ruler overlap](#)

Use the following CSS to ruler overlap properties.

```
`scss
.e-diagram .e-ruler-overlap {
background-color: red;
}
`
```

[Customizing the text edit](#)

Use the following CSS to customize the text edit properties.

```
`scss
.e-diagram .e-diagram-text-edit {
background: white;
border-color: red;
border-style: dashed;
border-width: 1px;
box-sizing: content-box;
color: black;
min-width: 50px;
}
`
```

[Customizing the text edit on selection](#)

Use the following CSS to customize the text edit on selection properties.

```
`scss
.e-diagram-text-edit::selection {
background: red;
color: green;
}
`
```


Ruler

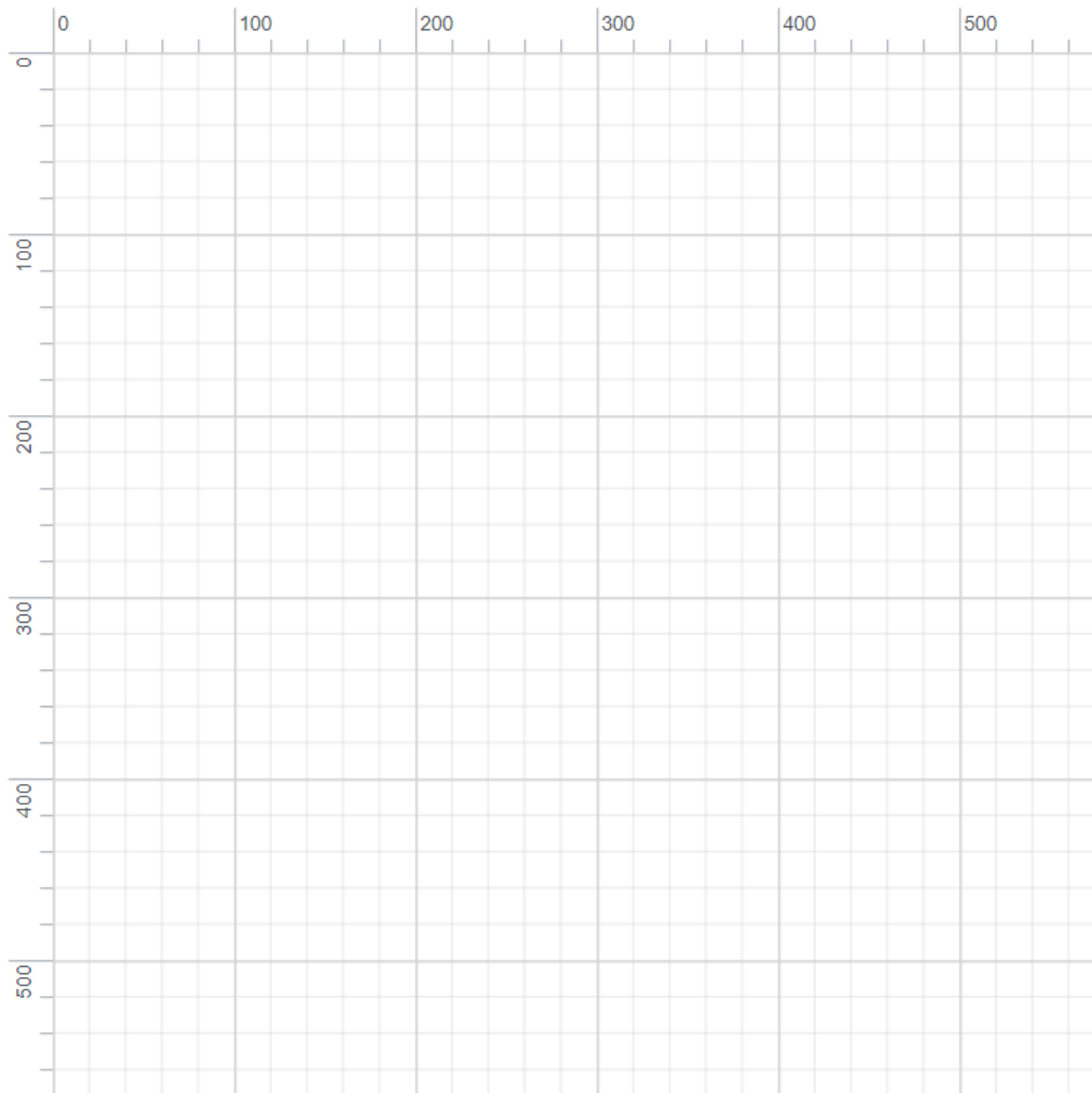
The Ruler provides a horizontal and vertical guide for measuring in the Diagram control. The Ruler can be used to measure the diagram objects, indicate positions, and align diagram elements. This is especially useful in creating scale models.

Adding Rulers to the Diagram

- The [rulerSettings](#) property is used to control the visibility and appearance of the ruler in the diagram.
- The RulerSettings [showRulers](#) property is used to show or hide the rulers in the diagram.
- The RulerSettings [horizontalRuler](#) and [verticalRuler](#) properties are used to customize the rulers appearance in the diagram.

RULER.CSHTML

```
<ejs-diagram id="diagram" width="100%" height="600px">  
  <e-diagram-rulersettings showRulers="true">  
  </e-diagram-rulersettings>  
</ejs-diagram>
```



Customizing the Ruler

By default, the ruler segments are arranged based on pixel values.

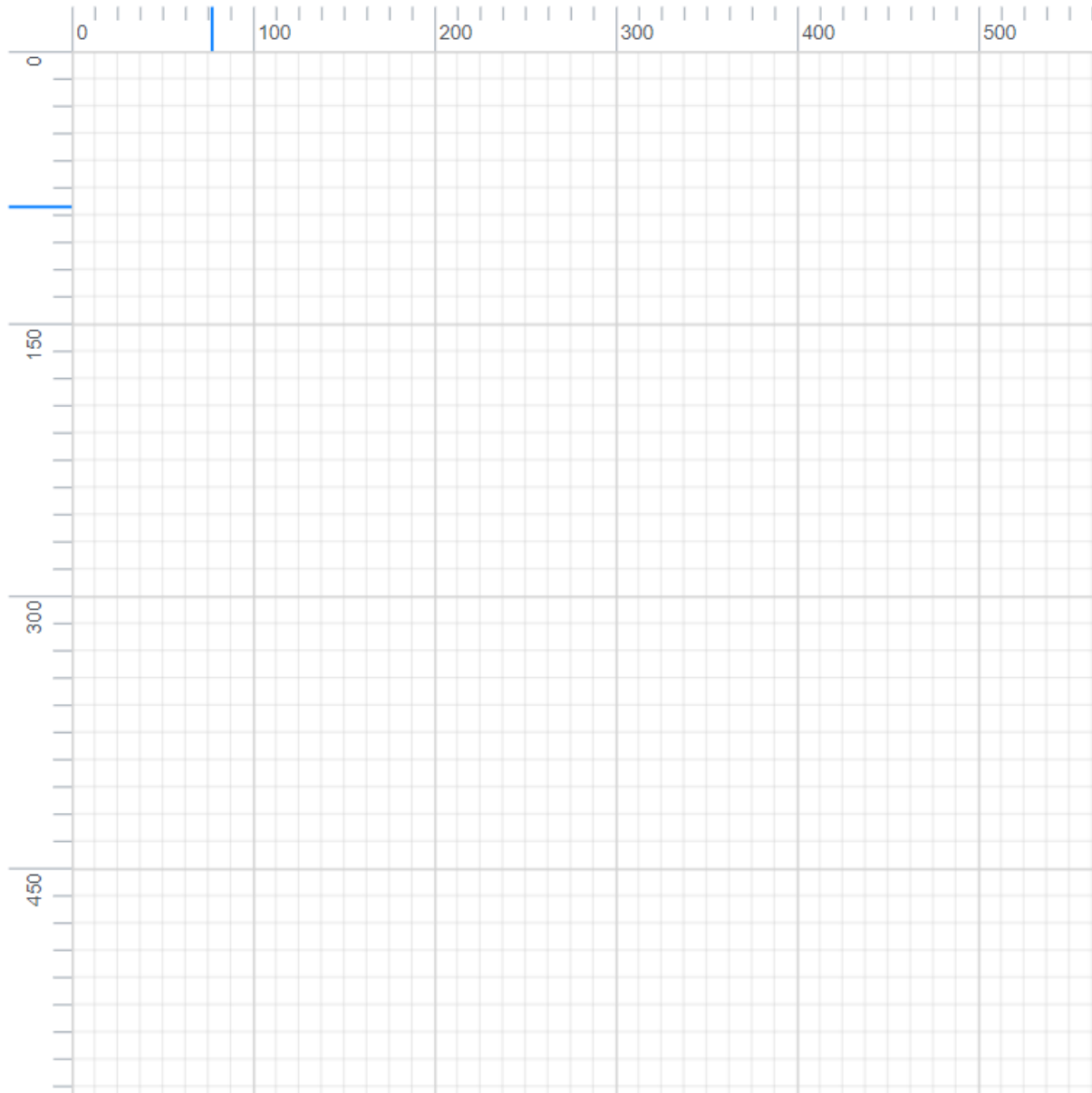
- The `HorizontalRuler`'s [interval](#) property allows to define the interval between ruler segments and the [segmentWidth](#) property allows to define the segment width of the ruler. Similarly, you can use the `VerticalRuler`'s [interval](#) and [segmentWidth](#) properties are used to define the interval and segment width of the vertical ruler.
- The `HorizontalRuler`'s [tickAlignment](#) property is used to align the ruler tick either left or right side of the ruler. The `VerticalRuler`'s [tickAlignment](#) property is used to align the ruler tick either top or bottom side of the ruler.
- The `HorizontalRuler`'s [arrangeTick](#) and `VerticalRuler`'s [arrangeTick](#) function is provided for the purpose of customizing the appearance of ruler ticks. It will be called for each tick rendering.

- The HorizontalRuler's [markerColor](#) and VerticalRuler's [markerColor](#) properties are used to define the ruler marker color and marker will be shown when performing the interaction in the diagram.

CUSTOMRULER.CSHTML

```
<ejs-diagram id="diagram" width="100%" height="600px">
  <e-diagram-rulersettings showRulers="true">
    <e-rulersettings-horizontalruler interval="8" segmentWidth="100"
thickness="25" tickAlignment="LeftOrTop"></e-rulersettings-horizontalruler>
    <e-rulersettings-verticalruler interval="10" segmentWidth="150"
thickness="35" tickAlignment="RightOrBottom"></e-rulersettings-
verticalruler>
  </e-diagram-rulersettings>
</ejs-diagram>
```

Note: The MarkerColor property can be customized using the [Marker](#) CSS style.



Layers in Diagram Control

Layer is used to organize related shapes on a diagram control. A layer is a named category of shapes. By assigning shapes to different layers, you can selectively view, remove, and lock different categories of shapes.

In diagram, [Layers](#) provide a way to change the properties of all shapes that have been assigned to that layer. The following properties can be set.

- Visible
- Lock
- Objects
- AddInfo

Visible

The layer's [visible](#) property is used to control the visibility of the elements assigned to the layer.

```
`typescript
```

```
import {
```

```
Diagram,
```

```
ConnectorModel,
```

```
NodeModel,
```

```
LayerModel
```

```
} from '@syncfusion/ej2-diagrams';
```

```
// A node is created and stored in nodes array.
```

```
let nodes: NodeModel[] = [{
```

```
id: 'node1',
```

```
width: 100,
```

```
height: 100,
```

```
offsetX: 100,
```

```
offsetY: 100,
```

```
annotations: [{
```

```
content: 'Default Shape'
```

```
}]
```

```
},
```

```
{
```

```
id: 'node2',
```

```
width: 100,
```

```
height: 100,
```

```
offsetX: 300,
```

```
offsetY: 100,
```

```
shape: {
```

```
type: 'Path',
```

```
data:
```

```
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L540.3643,' +
```

```
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,137.9336z'
```

```
},
```

```
annotations: [{
  content: 'Path Element'
}]
}
];
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];
// initialize diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  connectors: connectors,
  nodes: nodes,
  layers: [{
    id: 'layer1',
    visible: true,
    objects: ['node1']
  }
]
});
// render initialized diagram
diagram.appendTo('#element');
`
```

Lock

The layer's [lock](#) property is used to prevent or allow changes to the elements dimension and position.

`typescript

```
import {
  Diagram,
  ConnectorModel,
  NodeModel,
  LayerModel
} from '@syncfusion/ej2-diagrams';

// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Default Shape'
  }]
},
{
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  shape: {
    type: 'Path',
    data:
      'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L540.3643,194.9966L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,137.9336z'
  }
},
```

```
annotations: [{  
  content: 'Path Element'  
}]  
}  
];  
let connectors: ConnectorModel[] = [{  
  id: 'connector1',  
  type: 'Straight',  
  sourcePoint: {  
    x: 100,  
    y: 300  
  },  
  targetPoint: {  
    x: 200,  
    y: 400  
  },  
}];  
// initialize diagram component  
let diagram: Diagram = new Diagram({  
  width: '100%',  
  height: '600px',  
  connectors: connectors,  
  nodes: nodes,  
  layers: [{  
    id: 'layer1',  
    visible: true,  
    objects: ['node1'],  
    lock: true  
  },  
  {  
    id: 'layer2',  
    visible: true,  
    objects: ['node2'],
```



```
lock: false
}
]
});
// render initialized diagram
diagram.appendTo('#element');
`
```

Objects

The layer's [objects](#) property defines the diagram elements to the layer.

```
`typescript
import {
  Diagram,
  ConnectorModel,
  NodeModel,
  LayerModel
} from '@syncfusion/ej2-diagrams';
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Default Shape'
  }]
},
{
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
```

```
shape: {
  type: 'Path',
  data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
  annotations: [{
    content: 'Path Element'
  }]
}
];
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];
// initialize diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  connectors: connectors,
  nodes: nodes,
  layers: [{
    id: 'layer1',
    visible: true,
```

```
objects: ['node1', 'node2']
},
{
id: 'layer2',
visible: true,
objects: ['node2']
}
]
});
// render initialized diagram
diagram.appendTo('#element');
`
```

AddInfo

The [addInfo](#) property of layers allow you to maintain additional information to the layers.

```
`typescript
import {
Diagram,
ConnectorModel,
NodeModel,
LayerModel
} from '@syncfusion/ej2-diagrams';
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
id: 'node1',
width: 100,
height: 100,
offsetX: 100,
offsetY: 100,
annotations: [{
content: 'Default Shape'
}]
},
{

```

```
id: 'node2',
width: 100,
height: 100,
offsetX: 300,
offsetY: 100,
shape: {
  type: 'Path',
  data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
  content: 'Path Element'
}]
}
];
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];
let addInfo: Object = { Description: 'Layer1' };
// initialize Diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
```

```

height: '600px',
connectors: connectors,
nodes: nodes,
layers: [{
  id: 'layer1',
  visible: true,
  objects: ['node1', 'node2'],
  addInfo: addInfo
},
{
  id: 'layer2',
  visible: true,
  objects: ['node2']
}
]);
`

```

Add layer at runtime

Layers can be added at runtime by using the [addLayer](#) public method.

The layer's [ID](#) property defines the ID of the layer, and its further used to find the layer at runtime and do any customization.

```

`typescript
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
// add the layers to the existing diagram layer collection
diagram.addLayer({
  id: 'newlayer',
  objects: [],
  visible: true,
  lock: false,
  zIndex: -1
}, [{
  type: 'Straight',
  sourcePoint: {

```

```
x: 100,  
y: 300  
,  
targetPoint: {  
  x: 200,  
  y: 400  
}  
}]);  
,
```

Remove layer at runtime

Layers can be removed at runtime by using the [removeLayer](#) public method.

```
`typescript  
let diagramElement = document.getElementById('element');  
let diagram: Object[] = diagramElement.ej2_instances[0];  
// remove the diagram layers  
diagram.removeLayer([diagram.model.layers[i]]);  
,
```

moveObjects

Objects of the layers can be moved by using the [moveObjects](#) public method.

```
`typescript  
let diagramElement = document.getElementById('element');  
let diagram: Object[] = diagramElement.ej2_instances[0];  
// move the objects of diagram layers  
diagram.moveObjects(['connector1'], 'layer2');  
,
```

bringLayerForward

Layers can be moved forward at runtime by using the [bringLayerForward](#) public method.

```
`typescript  
let diagramElement = document.getElementById('element');  
let diagram: Object[] = diagramElement.ej2_instances[0];  
// move the layer forward  
diagram.bringLayerForward('layer1');  
,
```

sendLayerBackward

Layers can be moved backward at runtime by using the [sendLayerBackward](#) public method.

```
`typescript
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
// move the layer backward
diagram.sendLayerBackward('layer1');
`
```

cloneLayer

Layers can be cloned with its object by using the [cloneLayer](#) public method.

```
`typescript
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
// clone a layer with its object
diagram.cloneLayer('layer2');
`
```

getActiveLayer

To get the active layers back in diagram, use the [getActiveLayer](#) public method.

```
`typescript
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
// gets the active layer back
diagram.getActiveLayer();
`
```

setActiveLayer

Set the active layer by using the [setActiveLayer](#) public method.

```
`typescript
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
// set the active layer
//@param layerName defines the name of the layer which is to be active layer
diagram.setActiveLayer('layer2');
`
```

Context Menu in Diagram

<!-- markdownlint-disable MD010 -->

In graphical user interface (GUI), a context menu is a type of menu that appears when you perform right-click operation. Nested level of context menu items can be created. Diagram provides some in-built context menu items and allows to define custom menu items through the [contextMenuSettings](#) property.

Customize context menu

The [show](#) property helps to enable or disable the context menu. Diagram provides some default context menu items to ease the execution of some frequently used commands.

DEFAULT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "node2", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            nodes.Add(new DiagramNode() {
                Id = "node2",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 300,
```



```

        OffsetY = 300,
        Annotations = Node2
    });
    ViewBag.nodes = nodes;
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 } });
    ViewBag.connectors = Connectors;
    return View();
    }
}
}

```

Context menu can be defined for individual node with the desired context menu items.

- Apart from the default context menu items, define some additional context menu items. Those additional items have to be defined and added to the [items](#) property of the context menu.
- Set text and ID for context menu item using the context menu [text](#) and [ID](#) properties respectively.
- Set an image for the context menu item using the context menu url property.
- The `iconCss` property defines the class or multiple classes separated by a space for the menu item that is used to include an icon. Menu item can include font icon and sprite image.
- The `target` property used to set the target to show the menu item.
- The `separator` property defines the horizontal lines that are used to separate the menu items. You cannot select the separators. You can enable separators to group the menu items using the `separator` property.

ITEM.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.ComponentModel;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace sample1.Controllers
{
    public class NodeController: Controller
    {
        // GET: Node
        public ActionResult Node()
        {
            // Sets the Annotation for the Node
            List<DiagramNode> Nodes = new List<DiagramNode>();

            List<DiagramNodeAnnotation> Node1 = new
List<DiagramNodeAnnotation>();
            Node1.Add(new DiagramNodeAnnotation()

```

```

        {
            //Sets the offset for the content
            Content = "Node1",
            Hyperlink = new { link = "" },
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent"
            },
            //Sets the margin for the annotation
            Margin = new DiagramMargin() { Top = 10 },
            // Sets the horizontal alignment as left
            HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
            // Sets the vertical alignment as Center
            VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center
        });
        List<DiagramNodeAnnotation> Node2 = new
List<DiagramNodeAnnotation>();
        Node2.Add(new DiagramNodeAnnotation()
        {
            //Sets the offset for the content
            Content = "Node1",
            Style = new DiagramTextStyle()
            {
                Color = "black",
                Fill = "transparent"
            },
            //Sets the margin for the annotation
            Margin = new DiagramMargin() { Bottom = 10 },
            // Sets the horizontal alignment as left
            HorizontalAlignment =
Syncfusion.EJ2.Diagrams.HorizontalAlignment.Left,
            // Sets the vertical alignment as Center
            VerticalAlignment =
Syncfusion.EJ2.Diagrams.VerticalAlignment.Center
        });
        Nodes.Add(new DefaultNode()
        {
            Id = "Node1",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width = 100,
            // add the Annotation for the Node
            Annotations = Node1,
        });
        Nodes.Add(new DefaultNode()
        {
            Id = "Node2",
            OffsetY = 100,
            OffsetX = 100,
            Height = 100,
            Width = 100,
            // add the Annotation for the Node
            Annotations = Node2,
        });
    }
}

```

```

    }); // Sets the Annotation for the Connector
    List<DiagramConnectorAnnotation> Connector1 = new
List<DiagramConnectorAnnotation>();
    Connector1.Add(new DiagramConnectorAnnotation()
    {
        Content = "Connector1",
        // Sets the offset for the content
        Offset = 0,
        //Sets the margin for the annotation
        Margin = new DiagramMargin() { Top = 10 },
        Style = new DiagramTextStyle() { Color = "white", Fill =
"transparent" }
    });
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector()
    {
        Id = "connector1",
        SourceID = "Node1",
        TargetID = "Node2",
        // add the Annotation for the Connector
        Annotations = Connector1
    });
    ViewBag.nodes = Nodes;
    ViewBag.connectors = Connectors;
    List<contextMenuItems> item = new List<contextMenuItems>();
    item.Add(new contextMenuItems()
    {
        Id = "save",
        Text = "Save",
        Target = ".e-elementcontent",
        IconCss = "e-save"
    });
    item.Add(new contextMenuItems()
    {
        Id = "load",
        Text = "Load",
        Target = ".e-elementcontent",
        IconCss = "e-load"
    });
    item.Add(new contextMenuItems()
    {
        Id = "clear",
        Text = "Clear",
        Target = ".e-elementcontent",
        IconCss = "e-clear"
    });
    ViewBag.item = item;
    return View();
}

public class contextMenuItems
{
    [DefaultValue(null)]
    [HtmlAttributeName("text")]
    [JsonProperty("text")]
    public string Text
    {

```

```

        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("id")]
    [JsonProperty("id")]
    public string Id
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("target")]
    [JsonProperty("target")]
    public string Target
    {
        get;
        set;
    }
    [DefaultValue(null)]
    [HtmlAttributeName("iconCss")]
    [JsonProperty("iconCss")]
    public string IconCss
    {
        get;
        set;
    }
    }
}

```

To display the custom context menu items alone, set the `showCustomMenuOnly` property to true.

Template Support for Context menu

- Diagram provides template support for context menu. The context menu items can be customized by using the `contextMenuBeforeItemRender` event. The `contextMenuBeforeItemRender` event triggers while rendering each menu item.
- In the following sample, the menu item is rendered with key code for specified action in ContextMenu using the template. Here, the key code is specified for the cut and copy at right corner of the menu items by adding a span element in the `contextMenuBeforeItemRender` event.

EVENTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
    }
}

```

```

public ActionResult Nodes() {
    List < DiagramNode > nodes = new List < DiagramNode > ();
    List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
    Node1.Add(new DiagramNodeAnnotation() {
        Content = "node1", Style = new DiagramTextStyle() {
            Color = "White", StrokeColor = "None"
        }
    });
    List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
    Node2.Add(new DiagramNodeAnnotation() {
        Content = "node2", Style = new DiagramTextStyle() {
            Color = "White", StrokeColor = "None"
        }
    });
    nodes.Add(new DiagramNode() {
        Id = "node1",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 100,
        OffsetY = 100,
        Annotations = Node1
    });
    nodes.Add(new DiagramNode() {
        Id = "node2",
        Width = 100,
        Height = 100,
        Style = new NodeStyleNodes() {
            Fill = "darkcyan"
        },
        OffsetX = 300,
        OffsetY = 300,
        Annotations = Node2
    });
    ViewBag.nodes = nodes;
    List<DiagramConnector> Connectors = new
List<DiagramConnector>();
    Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 } });
    ViewBag.connectors = Connectors;
    return View();
}
}

```

Context menu events

You would be notified with events, when you try to open the context menu items `contextMenuOpen` and when you click the menu items `contextMenuClick`.

EVENTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > nodes = new List < DiagramNode > ();
            List < DiagramNodeAnnotation > Node1 = new List <
DiagramNodeAnnotation > ();
            Node1.Add(new DiagramNodeAnnotation() {
                Content = "node1", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            List < DiagramNodeAnnotation > Node2 = new List <
DiagramNodeAnnotation > ();
            Node2.Add(new DiagramNodeAnnotation() {
                Content = "node2", Style = new DiagramTextStyle() {
                    Color = "White", StrokeColor = "None"
                }
            });
            nodes.Add(new DiagramNode() {
                Id = "node1",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 100,
                OffsetY = 100,
                Annotations = Node1
            });
            nodes.Add(new DiagramNode() {
                Id = "node2",
                Width = 100,
                Height = 100,
                Style = new NodeStyleNodes() {
                    Fill = "darkcyan"
                },
                OffsetX = 300,
                OffsetY = 300,
                Annotations = Node2
            });
            ViewBag.nodes = nodes;
            List<DiagramConnector> Connectors = new
List<DiagramConnector>();
            Connectors.Add(new DiagramConnector() { Id = "connector",
SourcePoint = new DiagramPoint() { X = 100, Y = 100 }, TargetPoint = new
DiagramPoint() { X = 300, Y = 300 } });
            ViewBag.connectors = Connectors;
            return View();
        }
    }
}

```

```
}  
}
```

```
`javascript  
function contextMenuOpen() {  
    for (var item of args.items) {  
        if (item.text === 'delete') {  
            var diagram = document.getElementById("container").ej2_instances[0];  
            if (!diagram.selectedItems.nodes.length && !diagram.selectedItems.connectors.length) {  
                args.hiddenItems.push(item.id);  
            }  
        }  
    }  
}  
  
function contextMenuClick() {  
    if (args.item.id === 'delete') {  
        var diagram = document.getElementById("container").ej2_instances[0];  
        if ((diagram.selectedItems.nodes.length + diagram.selectedItems.connectors.length) > 0) {  
            diagram.cut();  
        }  
    }  
}  
`
```

Symbol Palette in Diagram

The **SymbolPalette** displays a collection of palettes. The palette shows a set of nodes and connectors. It allows to drag and drop the nodes and connectors into the diagram.

Create symbol palette

The [width](#) and [height](#) properties of the symbol palette allows to define the size of the symbol palette.

PALETTE.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using Syncfusion.EJ2.Diagrams;  
using System.Drawing;  
namespace EJ2MVCSampleBrowser.Controllers.Diagram {  
    public partial class DiagramController: Controller {
```

```
// GET: Nodes
public ActionResult Nodes() {
    return View();
}
}
```

Add palettes to SymbolPalette

A palette allows to display a group of related symbols and it textually annotates the group with its header. A [Palettes](#) can be added as a collection of symbol groups.

The collection of predefined symbols can be added in palettes using the [symbols](#) property.

To initialize a palette, define a JSON object with the property [ID](#) that is unique ID is set to the palettes.

SYMBOLS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > SymbolPalettee = new List <
DiagramNode > ();
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Terminator", Shape = new {
                    type = "Flow", shape = "Terminator"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Terminator", Shape = new {
                    type = "Flow", shape = "Terminator"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Process", Shape = new {
                    type = "Flow", shape = "Process"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Decision", Shape = new {
                    type = "Flow", shape = "Decision"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Document", Shape = new {
                    type = "Flow", shape = "Document"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
```



```

        Id = "PreDefinedProcess", Shape = new {
            type = "Flow", shape = "PreDefinedProcess"
        }
    });
    SymbolPalette.Add(new FlowShapes() {
        Id = "PaperTap", Shape = new {
            type = "Flow", shape = "PaperTap"
        }
    });
    SymbolPalette.Add(new FlowShapes() {
        Id = "DirectData", Shape = new {
            type = "Flow", shape = "DirectData"
        }
    });
    SymbolPalette.Add(new FlowShapes() {
        Id = "SequentialData", Shape = new {
            type = "Flow", shape = "SequentialData"
        }
    });
    List < DiagramNode > BasicNodes = new List <
DiagramNode > ();
    BasicNodes.Add(new DiagramNode() {
        Id = "Rectangle", Shape = new
DiagramBasicShape() {
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Rectangle
        }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Ellipse", Shape = new DiagramBasicShape()
{
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Ellipse
        }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Parallelogram", Shape = new
DiagramBasicShape() {
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Parallelogram
        }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Triangle", Shape = new DiagramBasicShape()
{
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Triangle
        }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Hexagon", Shape = new DiagramBasicShape()
{
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Hexagon
        }
    });
    BasicNodes.Add(new DiagramNode() {

```

```

        Id = "Pentagon", Shape = new DiagramBasicShape()
    {
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
        Shape = BasicShapes.Pentagon
    }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Cylinder", Shape = new DiagramBasicShape()
    {
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
        Shape = BasicShapes.Cylinder
    }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Star", Shape = new DiagramBasicShape() {
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
            Shape = BasicShapes.Star
        }
    });
    ViewBag.BasicShapes = BasicNodes;
    List < DiagramNode > SvgNodes = new List <
DiagramNode > ();

    SvgNodes.Add(new DiagramNode() {
        Id = "Native", Style = new NodeStyleNodes() {
            Fill = "None"
        }, Shape = new {
            type = "Native"
            content = "<g xmlns="
                http: //www.w3.org/2000/svg"><g
transform="translate(1 1)">
style="fill:#61443C;" d="M61.979,435.057c2.645-0.512,5.291-0.853,7.936-
1.109c-2.01,1.33-4.472,1.791-6.827,1.28
C62.726,435.13,62.354,435.072,61.979,435.057z"/>
style="fill:#61443C;" d="M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-
49.749-66.304 c-5.784-1.581-11.753-2.385-17.749-2.389c-2.425-0.028-
4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-23.296 c0.377-
47.729-34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-
2.731,23.296c2.405-0.312,4.829-0.455,7.253-0.427
C472.572,434.123,502.783,464.869,502.469,502.471z"/>
<path style="fill:#8B685A;" d="M476.869,502.471H7.536c-0.191-
32.558,22.574-60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-
1.28c1.707-0.085,3.413-0.171,5.12-0.171
c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-
1.288,3.414-3.341,3.879-5.685 c7.68-39.706,39.605-70.228,79.616-
76.117c4.325-0.616,8.687-0.929,13.056-0.939c13.281-
0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.
345-0.598,4.337-2.142,5.502-4.264 c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-
0.9,15.666-2.731,23.296 c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389C456.696,444.61,477.033,471.713,4
76.869,502.471 L476.869,502.471z"/>
<path style="fill:#66993E;"
d="M502.469,7.537c0,0-6.997,264.96-192.512,252.245c-20.217-1.549-40.166-
5.59-59.392-12.032 c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-

```

```
1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z"/>      <path
style="fill:#9ACA5C;" d="M476.784,10.183c-1.28,26.197-16.213,238.165-
166.827,249.6      c-20.217-1.549-40.166-5.59-59.392-12.032c-1.365-0.341-
2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144
C238.363,206.279,219.931,41.329,476.784,10.183z"/>      <path
style="fill:#66993E;" d="M206.192,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-21.505,7.427-
44.293,10.417-66.987,8.789C21.104,252.103,8.816,94.236,7.621,71.452c-0.085-
1.792-0.085-2.731-0.085-2.731
C222.747,86.129,211.653,216.689,206.192,246.727z"/>      <path
style="fill:#9ACA5C;" d="M180.336,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-13.351,4.412-
27.142,7.359-41.131,8.789C21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z"/>    </g>    <g>
      <path d="M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-
6.858-1.849-9.298l-8.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533
C157.701,425.773,159.872,426.673,162.136,426.671L162.136,426.671z"/>
      <path d="M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z"/>      <path
d="M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-
3.335,8.671-0.054,12.012l8.533,8.533      c3.341,3.281,8.701,3.256,12.012-
0.054c3.311-3.311,3.335-8.671,0.054-12.012L296.169,454.771z"/>
      <path d="M386.503,475.37c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
      <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-
2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-
3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z"/>      <path
d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z"/>      <path
d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z"/>      <path
d="M164.636,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z"/>
      <path d="M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z"/>
      <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-
12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z"/>      <path
d="M70.77,463.304l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271s3.1,5.28,6.065,6.065      c2.965,0.785,6.122-0.082,8.271-
2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z"/>      <path
d="M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-
```

```
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z"/>
  <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-
4.433,0.899-6.033,2.51-8.533,8.533      c-2.178,2.151-3.037,5.304-
2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-
0.073,8.262-2.25118.533-8.533      c1.601-1.6,2.5-3.77,2.5-
6.033C204.802,424.408,203.903,422.237,202.302,420.638L202.302,420.638z"/>
  <path d="M210.836,463.304c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-
2.965-0.082-6.122-2.27-8.271L210.836,463.304z"/>    <path
d="M343.836,454.7711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
  <path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-
8.671,0.054-12.0121-8.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>    <path
d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.0541-8.533,8.533c-
2.44,2.441-3.169,6.11-
1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z"/>
  <path d="M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z"/>
  <path
d="M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62
.581-8.828,88.583-26.796      c94.225-65.725,99.567-227.462,99.75-
234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-1.679-4.141-2.52-6.558-
2.308      C387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881      c-
4.581-7.691-9.996-14.854-16.147-21.358c8.023-38.158,0.241-77.939-21.57-
110.261C160.753,95.829,98.828,68.458,9.228,61.196      c-2.417-0.214-
4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665
c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-5.666-26.723-8.595-40.759-
8.6c-49.449,0.497-91.788,35.567-101.483,84.058      c-5.094-1.093-10.29-1.641-
15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-
0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-
2.499c1.6-1.6,2.499-3.771,2.499-6.035      c0.249-41.103-31.914-75.112-72.967-
77.154c0.65-4.78,0.975-9.598,0.975-14.421c0.914-45.674-28.469-86.455-72.083-
100.045      c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.13
8z M87.519,238.092      c-55.35-38.567-67.358-129.25-69.833-
158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-
162.473-140.577c-2.82-1.165-6.048-0.736-8.466,1.125      s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.77
6,161.392,144.874      C150.795,260.908,115.29,257.451,87.519,238.092z
M279.969,114.046c37.6-53.788,109.708-86.113,214.408-96.138      c-2.65,35.375-
17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-142.305,13.433
c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
```

```

6.76c0.404-3.022-0.835-6.031-3.25-7.892    c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z
M262.694,368.017c15.097-26.883,43.468-43.587,74.3-43.746
c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-0.857,14.351-
2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661    c3.899-
29.333,28.777-51.334,58.367-51.617c4-
0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467    c3.858-
2.519,6.531-6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-
70.192c16.045-0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,
368.017L262.694,368.017z"/>    </g></g>"
    }
    });
    SvgNodes.Add(new DiagramNode() {
        Id = "Sync", Style = new
NodeStyleNodes() {
        Fill = "None"
    }, Shape = new {
        type = "Native"
        content = "<g xmlns="
http: //www.w3.org/2000/svg">' +
        '<rect height="256" width="256"
fill="#34353F"/>' +
        '<path id="path1"
transform="rotate(0,128,128) translate(59,61.2230899333954)
scale(4.3125,4.3125) " fill="#FFFFFF"
d="M18.88501,23.042998L26.804993,23.042998 26.804993,30.969001
18.88501,30.969001z M9.4360352,23.042998L17.358032,23.042998
17.358032,30.969001 9.4360352,30.969001z
M0.014038086,23.042998L7.9360352,23.042998 7.9360352,30.969001
0.014038086,30.969001z M18.871033,13.609001L26.791016,13.609001
26.791016,21.535994 18.871033,21.535994z
M9.4219971,13.609001L17.342041,13.609001 17.342041,21.535994
9.4219971,21.535994z M0,13.609001L7.9219971,13.609001 7.9219971,21.535994
0,21.535994z M9.4219971,4.1859968L17.342041,4.1859968 17.342041,12.113998
9.4219971,12.113998z M0,4.1859968L7.9219971,4.1859968 7.9219971,12.113998
0,12.113998z M25.846008,0L32,5.2310026 26.773987,11.382995
20.619019,6.155998z"/>' +
        '</g>" ) });
    SvgNodes.Add(new DiagramNode() {
        Id = "Network", Style = new
NodeStyleNodes() {
        Fill = "None"
    }, Shape = new {
        type = "Native"
        content = "<g xmlns="
http:
//www.w3.org/2000/svg">' +
        '<rect height="256"
width="256" fill="#34353F"/>' +
        '<path id="path1"
transform="rotate(0,128,128) translate(59.1078108549118,59)
scale(4.3125,4.3125) " fill="#FFFFFF"
d="M12.12701,24.294998C12.75201,24.294998 13.258998,24.803009

```

13.258998,25.428009 13.258998,26.056 12.75201,26.563004 12.12701,26.563004
11.499019,26.563004 10.993007,26.056 10.993007,25.428009 10.993007,24.803009
11.499019,24.294998 12.12701,24.294998z
M7.9750035,24.294998C8.6010101,24.294998 9.1090057,24.803009
9.1090057,25.428009 9.1090057,26.056 8.6010101,26.563004 7.9750035,26.563004
7.3480199,26.563004 6.8399942,26.056 6.8399942,25.428009 6.8399942,24.803009
7.3480199,24.294998 7.9750035,24.294998z
M7.9750035,20.286011C8.6010101,20.286011 9.1090057,20.792999
9.1090057,21.419006 9.1090057,22.044006 8.6010101,22.552002
7.9750035,22.552002 7.3500035,22.552002 6.8420084,22.044006
6.8420084,21.419006 6.8420084,20.792999 7.3500035,20.286011
7.9750035,20.286011z
M18.499994,19.317001C18.313013,19.317001,18.156,19.472,18.156,19.656006L18.1
56,27.01001C18.156,27.195007,18.313013,27.350006,18.499994,27.350006L29.5219
93,27.350006C29.707998,27.350006,29.865988,27.195007,29.865988,27.01001L29.8
65988,19.656006C29.865988,19.472,29.707998,19.317001,29.521993,19.317001z
M17.243006,17.443008L30.778003,17.443008C31.425007,17.445007,31.947986,17.96
2006,31.950001,18.602997L31.950001,28.542007C31.947986,29.182999,31.425007,2
9.702011,30.778003,29.703003L25.654012,29.703003C25.511007,29.703003
25.399008,29.824997 25.413992,29.964996 25.430013,30.13501
25.452993,30.360001 25.477011,30.559998 25.506002,30.809998
25.727987,30.980011
25.976003,31.033997L27.756002,31.419006C27.907003,31.452011 28.015005,31.584
28.015005,31.738007 28.015005,31.883011 27.895986,32
27.74999,32L27.571005,32 20.450004,32 20.318016,32C20.171013,32
20.053001,31.883011 20.053001,31.738007 20.053001,31.585007
20.161003,31.452011
20.312004,31.419998L22.115989,31.033005C22.35601,30.98201
22.572014,30.815002 22.596,30.574005 22.616997,30.363007 22.636009,30.130997
22.648002,29.960007 22.658012,29.819 22.542015,29.70401
22.399986,29.70401L17.243006,29.703003C16.596002,29.702011,16.072992,29.1829
99,16.071008,28.542007L16.071008,18.602997C16.072992,17.962006,16.596002,17.
445007,17.243006,17.443008z M7.9750035,16.133011C8.6020172,16.133011
9.1100128,16.641006 9.1100128,17.268005 9.1100128,17.893997
8.6020172,18.402008 7.9750035,18.402008 7.3489964,18.402008
6.8410013,17.893997 6.8410013,17.268005 6.8410013,16.641006
7.3489964,16.133011 7.9750035,16.133011z
M24.027,13.762009C24.654014,13.762009 25.16201,14.270004 25.16201,14.895996
25.16201,15.522003 24.654014,16.029999 24.027,16.029999 23.400993,16.029999
22.892998,15.522003 22.892998,14.895996 22.892998,14.270004
23.400993,13.762009 24.027,13.762009z M24.027,9.6110077C24.653007,9.6110077
25.161003,10.119003 25.161003,10.74501 25.161003,11.37001
24.653007,11.878006 24.027,11.878006 23.402,11.878006 22.894005,11.37001
22.894005,10.74501 22.894005,10.119003 23.402,9.6110077 24.027,9.6110077z
M24.027,5.6000061C24.654014,5.6000061 25.16201,6.1080017 25.16201,6.7350006
25.16201,7.3610077 24.654014,7.8690033 24.027,7.8690033 23.400993,7.8690033
22.892998,7.3610077 22.892998,6.7350006 22.892998,6.1080017
23.400993,5.6000061 24.027,5.6000061z
M19.876001,5.6000061C20.503013,5.6000061 21.011009,6.1080017
21.011009,6.7350006 21.011009,7.3610077 20.503013,7.8690033
19.876001,7.8690033 19.249994,7.8690033 18.743006,7.3610077
18.743006,6.7350006 18.743006,6.1080017 19.249994,5.6000061
19.876001,5.6000061z
M2.4290157,1.8740082C2.2420037,1.8740082,2.0850215,2.029007,2.0850215,2.2140
045L2.0850215,9.5680084C2.0850215,9.753006,2.2420037,9.9069977,2.4290157,9.9
069977L13.451014,9.9069977C13.637995,9.9069977,13.795008,9.753006,13.795008,
9.5680084L13.795008,2.2140045C13.795008,2.029007,13.637995,1.8740082,13.4510

```

14,1.8740082z
M1.1730042,0L14.706996,0C15.353999,0.0019989014,15.877009,0.51899719,15.8789
93,1.1600037L15.878993,11.100006C15.877009,11.740005,15.353999,12.26001,14.7
06996,12.26001L9.5830047,12.26001C9.4399994,12.26001 9.3290069,12.382004
9.3420074,12.52301 9.3600128,12.692001 9.3829925,12.917999
9.4060028,13.117004 9.4349945,13.367004 9.6570099,13.53801
9.9049957,13.591003L11.684994,13.975998C11.835994,14.009003
11.945003,14.141998 11.945003,14.294998 11.945003,14.440002
11.826015,14.557007 11.679012,14.557007L11.499996,14.557007
4.3789966,14.557007 4.2470081,14.557007C4.1000049,14.557007
3.9819935,14.440002 3.9819937,14.294998 3.9819935,14.141998
4.0899952,14.009003
4.2409961,13.977005L6.0450113,13.589996C6.2860086,13.539001
6.501005,13.373001 6.5249918,13.130997 6.5460184,12.921005
6.5650003,12.688004 6.5769937,12.516998 6.5870035,12.376999
6.4710062,12.262009
6.3290079,12.262009L1.1730042,12.26001C0.52499391,12.26001,0.0020143806,11.7
40005,0,11.100006L0,1.1600037C0.0020143806,0.51899719,0.52499391,0.001998901
4,1.1730042,0z"/>' +

                                '</g>'
                                )
                                });
                                ViewBag.SvgShapes = SvgNodes;
                                List < DiagramNode > HtmlNodes = new

List < DiagramNode > ();

                                HtmlNodes.Add(new DiagramNode() {
                                    Id = "Button", Shape = new {
                                        type = "HTML", content =

                                height: 100 % ;
                                width: 100 % ;
                                "><button type="
                                button " style="
                                width: 100 % ;
                                overflow: hidden

                                }
                                });
                                HtmlNodes.Add(new DiagramNode() {
                                    Id = "Checkbox", Shape = new {

                                type = "HTML", content =

                                height: 50 % ;
                                width: 100 % ;
                                "><input type="
                                checkbox " value="
                                Yes " style="
                                width: 25 %

                                ">Yes</input></div>' +

                                '<div
                                style="height:50%;width:100%;"><input type="checkbox" value="No"
                                style="width:25%">No</input></div></div>'
                                }); HtmlNodes.Add(new

DiagramNode() {

                                Id = "Dropdown", Shape = new

{

```

```

type = "HTML", content =
"<<div style="
                                height: 100 % ;
                                width: 100 % ;
                                "><select style="
                                width: 100 %
"><option>SVG</option><option>Canvas</option></select></div>"
                                }
                                }); ViewBag.HtmlShapes =
HtmlNodes;
                                List < DiagramConnector >
connector = new List < DiagramConnector > (); connector.Add(new
DiagramConnector() {
                                Id = "link1", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() {
                                X = 0, Y = 0
                                }, TargetPoint = new
DiagramPoint() {
                                X = 40, Y = 40
                                }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
                                Shape =
DecoratorShapes.Arrow
                                }, Style = new
DiagramStrokeStyle() {
                                StrokeWidth = 2
                                }
                                }); connector.Add(new
DiagramConnector() {
                                Id = "link3", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() {
                                X = 0, Y = 0
                                }, TargetPoint = new
DiagramPoint() {
                                X = 40, Y = 40
                                }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
                                Shape =
DecoratorShapes.None
                                }
                                }); connector.Add(new
DiagramConnector() {
                                Id = "Link21", Type =
Segments.Straight, SourcePoint = new DiagramPoint() {
                                X = 0, Y = 0
                                }, TargetPoint = new
DiagramPoint() {
                                X = 40, Y = 40
                                }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
                                Shape =
DecoratorShapes.Arrow
                                }, Style = new
DiagramStrokeStyle() {
                                StrokeWidth = 2
                                }
                                }

```



```

}); connector.Add(new
DiagramConnector() {
    Id = "link23", Type =
Segments.Straight, SourcePoint = new DiagramPoint() {
    X = 0, Y = 0
}, TargetPoint = new
DiagramPoint() {
    X = 40, Y = 40
}, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
    Shape =
DecoratorShapes.None
}
}); connector.Add(new
DiagramConnector() {
    Id = "link33", Type =
Segments.Bezier, SourcePoint = new DiagramPoint() {
    X = 0, Y = 0
}, TargetPoint = new
DiagramPoint() {
    X = 40, Y = 40
}, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
    Shape =
DecoratorShapes.None
}
}); ViewBag.Connectors =
connector;
List < SymbolPalettePalette >
palettes = new List < SymbolPalettePalette > (); palettes.Add(new
SymbolPalettePalette() {
    Id = "flow", Expanded =
true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes"
}); palettes.Add(new
SymbolPalettePalette() {
    Id = "basic", Expanded =
true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes"
});
palettes.Add(new
SymbolPalettePalette() { Id="svg", Expanded = true, symbols = SvgNodes,
Title = "SVG Shapes" });
palettes.Add(new
SymbolPalettePalette() { Id="html", Expanded = true, symbols = HtmlNodes,
Title = "HTML Shapes" });
palettes.Add(new
SymbolPalettePalette() {
    Id = "connectors", Expanded
= true, Symbols = connector, IconCss = "e-ddb-icons e-connector", Title =
"Connectors"
}); ViewBag.palettes = palettes;
return View();
}
}

```

Customize the palette header

Palettes can be annotated with its header texts.

The [title](#) displayed as the header text of palette.

The [expanded](#) property of palette allows to expand/collapse its palette items.

The [height](#) property of palette sets the height of the symbol group.

The [iconCss](#) property sets the content of the symbol group.

The [description](#) defines the text to be displayed and how that is to be handled in `getSymbolInfo`.

Also, any HTML element into a palette header can be embedded by defining the `getSymbolInfo` property.

SYMBOLS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List < DiagramNode > SymbolPalettee = new List <
DiagramNode > ();
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Terminator", Shape = new {
                    type = "Flow", shape = "Terminator"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Terminator", Shape = new {
                    type = "Flow", shape = "Terminator"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Process", Shape = new {
                    type = "Flow", shape = "Process"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Decision", Shape = new {
                    type = "Flow", shape = "Decision"
                }
            });
            SymbolPalettee.Add(new FlowShapes() {
                Id = "Document", Shape = new {
                    type = "Flow", shape = "Document"
                }
            });
        }
    }
}
```

```

SymbolPalettee.Add(new FlowShapes() {
    Id = "PreDefinedProcess", Shape = new {
        type = "Flow", shape = "PreDefinedProcess"
    }
});
SymbolPalettee.Add(new FlowShapes() {
    Id = "PaperTap", Shape = new {
        type = "Flow", shape = "PaperTap"
    }
});
SymbolPalettee.Add(new FlowShapes() {
    Id = "DirectData", Shape = new {
        type = "Flow", shape = "DirectData"
    }
});
SymbolPalettee.Add(new FlowShapes() {
    Id = "SequentialData", Shape = new {
        type = "Flow", shape = "SequentialData"
    }
});
List < DiagramNode > BasicNodes = new List <
DiagramNode > ();
BasicNodes.Add(new DiagramNode() {
    Id = "Rectangle", Shape = new
DiagramBasicShape() {
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Rectangle
    }
});
BasicNodes.Add(new DiagramNode() {
    Id = "Ellipse", Shape = new DiagramBasicShape()
{
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Ellipse
    }
});
BasicNodes.Add(new DiagramNode() {
    Id = "Parallelogram", Shape = new
DiagramBasicShape() {
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Parallelogram
    }
});
BasicNodes.Add(new DiagramNode() {
    Id = "Triangle", Shape = new DiagramBasicShape()
{
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Triangle
    }
});
BasicNodes.Add(new DiagramNode() {
    Id = "Hexagon", Shape = new DiagramBasicShape()
{
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
Shape = BasicShapes.Hexagon
    }
});

```

```

        BasicNodes.Add(new DiagramNode() {
            Id = "Pentagon", Shape = new DiagramBasicShape()
        {
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
            Shape = BasicShapes.Pentagon
        }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Cylinder", Shape = new DiagramBasicShape()
    {
        Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
        Shape = BasicShapes.Cylinder
    }
    });
    BasicNodes.Add(new DiagramNode() {
        Id = "Star", Shape = new DiagramBasicShape() {
            Type = Syncfusion.EJ2.Diagrams.Shapes.Basic,
            Shape = BasicShapes.Star
        }
    });
    ViewBag.BasicShapes = BasicNodes;
    List < DiagramNode > SvgNodes = new List <
DiagramNode > ();

    SvgNodes.Add(new DiagramNode() {
        Id = "Native", Style = new NodeStyleNodes() {
            Fill = "None"
        }, Shape = new {
            type = "Native"
            content = "<g xmlns="
            http: //www.w3.org/2000/svg"><g
transform="translate(1 1)">
style="fill:#61443C;" d="M61.979,435.057c2.645-0.512,5.291-0.853,7.936-
1.109c-2.01,1.33-4.472,1.791-6.827,1.28
C62.726,435.13,62.354,435.072,61.979,435.057z"/>
style="fill:#61443C;" d="M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-
49.749-66.304 c-5.784-1.581-11.753-2.385-17.749-2.389c-2.425-0.028-
4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-23.296 c0.377-
47.729-34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-
2.731,23.296c2.405-0.312,4.829-0.455,7.253-0.427
C472.572,434.123,502.783,464.869,502.469,502.471z"/>
<path style="fill:#8B685A;" d="M476.869,502.471H7.536c-0.191-
32.558,22.574-60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-
1.28c1.707-0.085,3.413-0.171,5.12-0.171
c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-
1.288,3.414-3.341,3.879-5.685 c7.68-39.706,39.605-70.228,79.616-
76.117c4.325-0.616,8.687-0.929,13.056-0.939c13.281-
0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.
345-0.598,4.337-2.142,5.502-4.264 c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-
0.9,15.666-2.731,23.296 c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389C456.696,444.61,477.033,471.713,4
76.869,502.471 L476.869,502.471z"/>
<path style="fill:#66993E;"
d="M502.469,7.537c0,0-6.997,264.96-192.512,252.245c-20.217-1.549-40.166-

```

```
5.59-59.392-12.032      c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-
1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z"/>      <path
style="fill:#9ACA5C;" d="M476.784,10.183c-1.28,26.197-16.213,238.165-
166.827,249.6      c-20.217-1.549-40.166-5.59-59.392-12.032c-1.365-0.341-
2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144
C238.363,206.279,219.931,41.329,476.784,10.183z"/>      <path
style="fill:#66993E;" d="M206.192,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-21.505,7.427-
44.293,10.417-66.987,8.789C21.104,252.103,8.816,94.236,7.621,71.452c-0.085-
1.792-0.085-2.731-0.085-2.731
C222.747,86.129,211.653,216.689,206.192,246.727z"/>      <path
style="fill:#9ACA5C;" d="M180.336,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-13.351,4.412-
27.142,7.359-41.131,8.789C21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z"/>      </g>      <g>
      <path d="M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-
6.858-1.849-9.298l-8.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533
C157.701,425.773,159.872,426.673,162.136,426.671z"/>
      <path d="M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z"/>      <path
d="M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-
3.335,8.671-0.054,12.012l8.533,8.533      c3.341,3.281,8.701,3.256,12.012-
0.054c3.311-3.311,3.335-8.671,0.054-12.012L296.169,454.771z"/>
      <path d="M386.503,475.37c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
      <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-
2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-
3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z"/>      <path
d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z"/>      <path
d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z"/>      <path
d="M164.636,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771z"/>
      <path d="M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171z"/>
      <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-
12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z"/>      <path
d="M70.77,463.304l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271s3.1,5.28,6.065,6.065      c2.965,0.785,6.122-0.082,8.271-
2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304z"/>      <path
```

```
d="M121.97,446.238L121.97,446.238z"/>
<path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-
4.433,0.899-6.033,2.51-8.533,8.533 c-2.178,2.151-3.037,5.304-
2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-
0.073,8.262-2.25118.533-8.533 c1.601-1.6,2.5-3.77,2.5-
6.033C204.802,424.408,203.903,422.237,202.302,420.638L202.302,420.638z"/>
<path d="M210.836,463.304c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-
2.965-0.082-6.122-2.27-8.271L210.836,463.304z"/>
<path
d="M343.836,454.7711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
<path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-
8.671,0.054-12.01218.533-8.533 c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>
<path
d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012 s-8.671-3.335-12.012-0.0541-8.533,8.533c-
2.44,2.441-3.169,6.11-
1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z"/>
<path d="M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012 s-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z"/>
<path
d="M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62
.581-8.828,88.583-26.796 c94.225-65.725,99.567-227.462,99.75-
234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-1.679-4.141-2.52-6.558-
2.308 c387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881 c-
4.581-7.691-9.996-14.854-16.147-21.358c8.023-38.158,0.241-77.939-21.57-
110.261C160.753,95.829,98.828,68.458,9.228,61.196 c-2.417-0.214-
4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665
c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-5.666-26.723-8.595-40.759-
8.6c-49.449,0.497-91.788,35.567-101.483,84.058 c-5.094-1.093-10.29-1.641-
15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-
0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-
2.499c1.6-1.6,2.499-3.771,2.499-6.035 c0.249-41.103-31.914-75.112-72.967-
77.154c0.65-4.78,0.975-9.598,0.975-14.421c0.914-45.674-28.469-86.455-72.083-
100.045 c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.13
8z M87.519,238.092 c-55.35-38.567-67.358-129.25-69.833-
158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-
162.473-140.577c-2.82-1.165-6.048-0.736-8.466,1.125 s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.77
6,161.392,144.874 C150.795,260.908,115.29,257.451,87.519,238.092z
M279.969,114.046c37.6-53.788,109.708-86.113,214.408-96.138 c-2.65,35.375-
17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-142.305,13.433
```

```

c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
6.76c0.404-3.022-0.835-6.031-3.25-7.892    c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z
M262.694,368.017c15.097-26.883,43.468-43.587,74.3-43.746
c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-0.857,14.351-
2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661    c3.899-
29.333,28.777-51.334,58.367-51.617c4-
0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467    c3.858-
2.519,6.531-6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-
70.192c16.045-0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,
368.017L262.694,368.017z"/>    </g></g>"
    }
    });
    SvgNodes.Add(new DiagramNode() {
        Id = "Sync", Style = new
NodeStyleNodes() {
        Fill = "None"
    }, Shape = new {
        type = "Native"
        content = "<g xmlns="
http: //www.w3.org/2000/svg">' +
        '<rect height="256" width="256"
fill="#34353F"/>' +
        '<path id="path1"
transform="rotate(0,128,128) translate(59,61.2230899333954)
scale(4.3125,4.3125) " fill="#FFFFFF"
d="M18.88501,23.042998L26.804993,23.042998 26.804993,30.969001
18.88501,30.969001z M9.4360352,23.042998L17.358032,23.042998
17.358032,30.969001 9.4360352,30.969001z
M0.014038086,23.042998L7.9360352,23.042998 7.9360352,30.969001
0.014038086,30.969001z M18.871033,13.609001L26.791016,13.609001
26.791016,21.535994 18.871033,21.535994z
M9.4219971,13.609001L17.342041,13.609001 17.342041,21.535994
9.4219971,21.535994z M0,13.609001L7.9219971,13.609001 7.9219971,21.535994
0,21.535994z M9.4219971,4.1859968L17.342041,4.1859968 17.342041,12.113998
9.4219971,12.113998z M0,4.1859968L7.9219971,4.1859968 7.9219971,12.113998
0,12.113998z M25.846008,0L32,5.2310026 26.773987,11.382995
20.619019,6.155998z"/>' +
        '</g>" ) });
    SvgNodes.Add(new DiagramNode() {
        Id = "Network", Style = new
NodeStyleNodes() {
        Fill = "None"
    }, Shape = new {
        type = "Native"
        content = "<g xmlns="
http:
//www.w3.org/2000/svg">' +
        '<rect height="256"
width="256" fill="#34353F"/>' +
        '<path id="path1"
transform="rotate(0,128,128) translate(59.1078108549118,59)
scale(4.3125,4.3125) " fill="#FFFFFF"

```

d="M12.12701,24.294998C12.75201,24.294998 13.258998,24.803009
13.258998,25.428009 13.258998,26.056 12.75201,26.563004 12.12701,26.563004
11.499019,26.563004 10.993007,26.056 10.993007,25.428009 10.993007,24.803009
11.499019,24.294998 12.12701,24.294998z
M7.9750035,24.294998C8.6010101,24.294998 9.1090057,24.803009
9.1090057,25.428009 9.1090057,26.056 8.6010101,26.563004 7.9750035,26.563004
7.3480199,26.563004 6.8399942,26.056 6.8399942,25.428009 6.8399942,24.803009
7.3480199,24.294998 7.9750035,24.294998z
M7.9750035,20.286011C8.6010101,20.286011 9.1090057,20.792999
9.1090057,21.419006 9.1090057,22.044006 8.6010101,22.552002
7.9750035,22.552002 7.3500035,22.552002 6.8420084,22.044006
6.8420084,21.419006 6.8420084,20.792999 7.3500035,20.286011
7.9750035,20.286011z
M18.499994,19.317001C18.313013,19.317001,18.156,19.472,18.156,19.656006L18.1
56,27.01001C18.156,27.195007,18.313013,27.350006,18.499994,27.350006L29.5219
93,27.350006C29.707998,27.350006,29.865988,27.195007,29.865988,27.01001L29.8
65988,19.656006C29.865988,19.472,29.707998,19.317001,29.521993,19.317001z
M17.243006,17.443008L30.778003,17.443008C31.425007,17.445007,31.947986,17.96
2006,31.950001,18.602997L31.950001,28.542007C31.947986,29.182999,31.425007,2
9.702011,30.778003,29.703003L25.654012,29.703003C25.511007,29.703003
25.399008,29.824997 25.413992,29.964996 25.430013,30.13501
25.452993,30.360001 25.477011,30.559998 25.506002,30.809998
25.727987,30.980011
25.976003,31.033997L27.756002,31.419006C27.907003,31.452011 28.015005,31.584
28.015005,31.738007 28.015005,31.883011 27.895986,32
27.74999,32L27.571005,32 20.450004,32 20.318016,32C20.171013,32
20.053001,31.883011 20.053001,31.738007 20.053001,31.585007
20.161003,31.452011
20.312004,31.419998L22.115989,31.033005C22.35601,30.98201
22.572014,30.815002 22.596,30.574005 22.616997,30.363007 22.636009,30.130997
22.648002,29.960007 22.658012,29.819 22.542015,29.70401
22.399986,29.70401L17.243006,29.703003C16.596002,29.702011,16.072992,29.1829
99,16.071008,28.542007L16.071008,18.602997C16.072992,17.962006,16.596002,17.
445007,17.243006,17.443008z M7.9750035,16.133011C8.6020172,16.133011
9.1100128,16.641006 9.1100128,17.268005 9.1100128,17.893997
8.6020172,18.402008 7.9750035,18.402008 7.3489964,18.402008
6.8410013,17.893997 6.8410013,17.268005 6.8410013,16.641006
7.3489964,16.133011 7.9750035,16.133011z
M24.027,13.762009C24.654014,13.762009 25.16201,14.270004 25.16201,14.895996
25.16201,15.522003 24.654014,16.029999 24.027,16.029999 23.400993,16.029999
22.892998,15.522003 22.892998,14.895996 22.892998,14.270004
23.400993,13.762009 24.027,13.762009z M24.027,9.6110077C24.653007,9.6110077
25.161003,10.119003 25.161003,10.74501 25.161003,11.37001
24.653007,11.878006 24.027,11.878006 23.402,11.878006 22.894005,11.37001
22.894005,10.74501 22.894005,10.119003 23.402,9.6110077 24.027,9.6110077z
M24.027,5.6000061C24.654014,5.6000061 25.16201,6.1080017 25.16201,6.7350006
25.16201,7.3610077 24.654014,7.8690033 24.027,7.8690033 23.400993,7.8690033
22.892998,7.3610077 22.892998,6.7350006 22.892998,6.1080017
23.400993,5.6000061 24.027,5.6000061z
M19.876001,5.6000061C20.503013,5.6000061 21.011009,6.1080017
21.011009,6.7350006 21.011009,7.3610077 20.503013,7.8690033
19.876001,7.8690033 19.249994,7.8690033 18.743006,7.3610077
18.743006,6.7350006 18.743006,6.1080017 19.249994,5.6000061
19.876001,5.6000061z
M2.4290157,1.8740082C2.2420037,1.8740082,2.0850215,2.029007,2.0850215,2.2140
045L2.0850215,9.5680084C2.0850215,9.753006,2.2420037,9.9069977,2.4290157,9.9
069977L13.451014,9.9069977C13.637995,9.9069977,13.795008,9.753006,13.795008,


```

9.5680084L13.795008,2.2140045C13.795008,2.029007,13.637995,1.8740082,13.4510
14,1.8740082z
M1.1730042,0L14.706996,0C15.353999,0.0019989014,15.877009,0.51899719,15.8789
93,1.1600037L15.878993,11.100006C15.877009,11.740005,15.353999,12.26001,14.7
06996,12.26001L9.5830047,12.26001C9.4399994,12.26001 9.3290069,12.382004
9.3420074,12.52301 9.3600128,12.692001 9.3829925,12.917999
9.4060028,13.117004 9.4349945,13.367004 9.6570099,13.53801
9.9049957,13.591003L11.684994,13.975998C11.835994,14.009003
11.945003,14.141998 11.945003,14.294998 11.945003,14.440002
11.826015,14.557007 11.679012,14.557007L11.499996,14.557007
4.3789966,14.557007 4.2470081,14.557007C4.1000049,14.557007
3.9819935,14.440002 3.9819937,14.294998 3.9819935,14.141998
4.0899952,14.009003
4.2409961,13.977005L6.0450113,13.589996C6.2860086,13.539001
6.501005,13.373001 6.5249918,13.130997 6.5460184,12.921005
6.5650003,12.688004 6.5769937,12.516998 6.5870035,12.376999
6.4710062,12.262009
6.3290079,12.262009L1.1730042,12.26001C0.52499391,12.26001,0.0020143806,11.7
40005,0,11.100006L0,1.1600037C0.0020143806,0.51899719,0.52499391,0.001998901
4,1.1730042,0z"/>' +

                                '</g>'
                                )
                                });
                                ViewBag.SvgShapes = SvgNodes;
                                List < DiagramNode > HtmlNodes = new

List < DiagramNode > ();

                                HtmlNodes.Add(new DiagramNode() {
                                    Id = "Button", Shape = new {
                                        type = "HTML", content =

                                height: 100 % ;
                                width: 100 % ;
                                "><button type="
                                button " style="
                                width: 100 % ;
                                overflow: hidden

                                }
                                });
                                HtmlNodes.Add(new DiagramNode() {
                                    Id = "Checkbox", Shape = new

                                type = "HTML", content =

                                height: 50 % ;
                                width: 100 % ;
                                "><input type="
                                checkbox " value="
                                Yes " style="
                                width: 25 %

                                ">Yes</input></div>' +

                                '<div
                                style="height:50%;width:100%;"><input type="checkbox" value="No"
                                style="width:25%">No</input></div></div>'

                                }); HtmlNodes.Add(new
DiagramNode() {

```

```

Id = "Dropdown", Shape = new
{
    type = "HTML", content =
    "<<div style="
    height: 100 % ;
    width: 100 % ;
    "><select style="
    width: 100 %
    "><option>SVG</option><option>Canvas</option></select></div>"
    }
}); ViewBag.HtmlShapes =
HtmlNodes;
List < DiagramConnector >
connector = new List < DiagramConnector > (); connector.Add(new
DiagramConnector() {
    Id = "link1", Type =
    Segments.Orthogonal, SourcePoint = new DiagramPoint() {
        X = 0, Y = 0
    }, TargetPoint = new
    DiagramPoint() {
        X = 40, Y = 40
    }, TargetDecorator = new
    ConnectorTargetDecoratorConnectors() {
        Shape =
        DecoratorShapes.Arrow
    }, Style = new
    DiagramStrokeStyle() {
        StrokeWidth = 2
    }
}); connector.Add(new
DiagramConnector() {
    Id = "link3", Type =
    Segments.Orthogonal, SourcePoint = new DiagramPoint() {
        X = 0, Y = 0
    }, TargetPoint = new
    DiagramPoint() {
        X = 40, Y = 40
    }, TargetDecorator = new
    ConnectorTargetDecoratorConnectors() {
        Shape =
        DecoratorShapes.None
    }
}); connector.Add(new
DiagramConnector() {
    Id = "Link21", Type =
    Segments.Straight, SourcePoint = new DiagramPoint() {
        X = 0, Y = 0
    }, TargetPoint = new
    DiagramPoint() {
        X = 40, Y = 40
    }, TargetDecorator = new
    ConnectorTargetDecoratorConnectors() {
        Shape =
        DecoratorShapes.Arrow
    }, Style = new
    DiagramStrokeStyle() {
        StrokeWidth = 2
    }
});

```

```

    }
    }); connector.Add(new
DiagramConnector() {
    Id = "link23", Type =
Segments.Straight, SourcePoint = new DiagramPoint() {
    X = 0, Y = 0
    }, TargetPoint = new
DiagramPoint() {
    X = 40, Y = 40
    }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
    Shape =
DecoratorShapes.None
    }
    }); connector.Add(new
DiagramConnector() {
    Id = "link33", Type =
Segments.Bezier, SourcePoint = new DiagramPoint() {
    X = 0, Y = 0
    }, TargetPoint = new
DiagramPoint() {
    X = 40, Y = 40
    }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() {
    Shape =
DecoratorShapes.None
    }
    }); ViewBag.Connectors =
connector;
List < SymbolPalettePalette >
palettes = new List < SymbolPalettePalette > (); palettes.Add(new
SymbolPalettePalette() {
    Id = "flow", Expanded =
true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes"
    }); palettes.Add(new
SymbolPalettePalette() {
    Id = "basic", Expanded =
true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes"
    });
palettes.Add(new
SymbolPalettePalette() { Id="svg", Expanded = true, symbols = SvgNodes,
Title = "SVG Shapes" });
palettes.Add(new
SymbolPalettePalette() { Id="html", Expanded = true, symbols = HtmlNodes,
Title = "HTML Shapes" });
palettes.Add(new
SymbolPalettePalette() {
    Id = "connectors", Expanded
= true, Symbols = connector, IconCss = "e-ddb-icons e-connector", Title =
"Connectors"
    }); ViewBag.palettes = palettes;
return View();
    }
}
}

```

```
`javascript
function getSymbolInfo(symbol) {
    if (symbol['text'] !== undefined) {
        return {
            width: 75,
            height: 40,
            //Add or Remove the Text for Symbol palette item.
            description: {
                //Defines the symbol description
                text: symbol['text'],
                //Defines how to handle the text when its size exceeds the given symbol size
                overflow: 'Wrap'
            }
        };
    }
    return {
        width: 75,
        height: 40,
        description: {
            text: symbol.shape['shape']
        }
    };
}
```

Restrict expansion of the palette panel

The symbol palette panel can be restricted from getting expanded. The `cancel` argument of the `paletteExpanding` property defines whether the palette's panel should be expanded or collapsed. By default, the panel is expanded. This restriction can be done for each of the palettes in the symbol palette as desired. In the following code example the basic shapes palette is restricted from getting collapsed whereas the swimlane shapes palette can be expanded or collapsed.

RESTRICT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```

using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
            SymbolPalettee.Add(new FlowShapes() { Id = "Process", Shape = new
{ type = "Flow", shape = "Process" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Document", Shape =
new { type = "Flow", shape = "Document" } });
            List<DiagramNode> BasicNodes = new List<DiagramNode>();
            BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Rectangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Ellipse } });
            BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Hexagon } });
            ViewBag.BasicShapes = BasicNodes;
            List<SymbolPalettePalette> palettes = new
List<SymbolPalettePalette>();
            palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
= true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes" });
            palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
= true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes" });
            ViewBag.palettes = palettes;
            ViewBag.paletteExpanding = "paletteExpanding";
            return View();
        }
    }
}

```

```

`javascript

```

```

function paletteExpanding(args) {
    if(args.palette.id === 'basic') {
        // Basic shapes panel does not collapse
        args.cancel = true;
    } else {
        // Swimlane shapes panel collapse and expand
        args.cancel = false;
    }
}

```

Stretch the symbols into the palette

The [fit](#) property defines whether the symbol has to be fit inside the size, that is defined by the symbol palette. For example, when you resize the rectangle in the symbol, ratio of the rectangle size has to be maintained rather changing into square shape.

FIT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
            SymbolPalettee.Add(new FlowShapes() { Id = "Terminator", Shape =
new { type = "Flow", shape = "Terminator" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Terminator", Shape =
new { type = "Flow", shape = "Terminator" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Process", Shape = new
{ type = "Flow", shape = "Process" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Decision", Shape =
new { type = "Flow", shape = "Decision" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Document", Shape =
new { type = "Flow", shape = "Document" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "PreDefinedProcess",
Shape = new { type = "Flow", shape = "PreDefinedProcess" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "PaperTap", Shape =
new { type = "Flow", shape = "PaperTap" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "DirectData", Shape =
new { type = "Flow", shape = "DirectData" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "SequentialData",
Shape = new { type = "Flow", shape = "SequentialData" } });
            List<DiagramNode> BasicNodes = new List<DiagramNode>();
            BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Rectangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Ellipse } });
            BasicNodes.Add(new DiagramNode() { Id = "Parallelogram", Shape =
new DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape
= BasicShapes.Parallelogram } });
            BasicNodes.Add(new DiagramNode() { Id = "Triangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Triangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Hexagon } });
            BasicNodes.Add(new DiagramNode() { Id = "Pentagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Pentagon } });
        }
    }
}
```

```

        BasicNodes.Add(new DiagramNode() { Id = "Cylinder", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Cylinder } });
        BasicNodes.Add(new DiagramNode() { Id = "Star", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Star } });
        ViewBag.BasicShapes = BasicNodes;
        List<DiagramConnector> connector = new List<DiagramConnector>();
        connector.Add(new DiagramConnector() { Id = "link1", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.Arrow },
Style = new DiagramStrokeStyle() { StrokeWidth = 2 } });
        connector.Add(new DiagramConnector() { Id = "link3", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        connector.Add(new DiagramConnector() { Id = "Link21", Type =
Segments.Straight, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.Arrow },
Style = new DiagramStrokeStyle() { StrokeWidth = 2 } });
        connector.Add(new DiagramConnector() { Id = "link23", Type =
Segments.Straight, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        connector.Add(new DiagramConnector() { Id = "link33", Type =
Segments.Bezier, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        ViewBag.Connectors = connector;
        List<SymbolPalettePalette> palettes = new
List<SymbolPalettePalette>();
        palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
= true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
= true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "connectors",
Expanded = true, Symbols = connector, IconCss = "e-ddb-icons e-connector",
Title = "Connectors" });
        ViewBag.palettes = palettes;
        ViewBag.getNodeDefaults = "getNodeDefaults";
        ViewBag.getSymbolInfo = "getSymbolInfo";
        return View();
    }
}

```

```
`javascript
```

```
function getSymbolInfo(symbol) {
return { fit: true };
}
```

Add/Remove symbols to palette at runtime

- Symbols can be added to palette at runtime by using public method, `addPalettetItem`.
- Symbols can be removed from palette at runtime by using public method, `removePalettetItem`.

Customize the size of symbols

The size of the individual symbol can be customized. The [symbolWidth](#) and [symbolHeight](#) properties of node enables to define the size of the symbols.

MARGIN.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
            SymbolPalettee.Add(new FlowShapes() { Id = "Terminator", Shape =
new { type = "Flow", shape = "Terminator" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Terminator", Shape =
new { type = "Flow", shape = "Terminator" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Process", Shape = new
{ type = "Flow", shape = "Process" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Decision", Shape =
new { type = "Flow", shape = "Decision" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Document", Shape =
new { type = "Flow", shape = "Document" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "PreDefinedProcess",
Shape = new { type = "Flow", shape = "PreDefinedProcess" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "PaperTap", Shape =
new { type = "Flow", shape = "PaperTap" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "DirectData", Shape =
new { type = "Flow", shape = "DirectData" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "SequentialData",
Shape = new { type = "Flow", shape = "SequentialData" } });
            List<DiagramNode> BasicNodes = new List<DiagramNode>();
            BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Rectangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Ellipse } });
            BasicNodes.Add(new DiagramNode() { Id = "Parallelogram", Shape =
new DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Parallelogram } });
            BasicNodes.Add(new DiagramNode() { Id = "Triangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Triangle } });
        }
    }
}
```



```

        BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Hexagon } });
        BasicNodes.Add(new DiagramNode() { Id = "Pentagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Pentagon } });
        BasicNodes.Add(new DiagramNode() { Id = "Cylinder", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Cylinder } });
        BasicNodes.Add(new DiagramNode() { Id = "Star", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Star } });
        ViewBag.BasicShapes = BasicNodes;
        List<DiagramConnector> connector = new List<DiagramConnector>();
        connector.Add(new DiagramConnector() { Id = "link1", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.Arrow },
Style = new DiagramStrokeStyle() { StrokeWidth = 2 } });
        connector.Add(new DiagramConnector() { Id = "link3", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        connector.Add(new DiagramConnector() { Id = "Link21", Type =
Segments.Straight, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.Arrow },
Style = new DiagramStrokeStyle() { StrokeWidth = 2 } });
        connector.Add(new DiagramConnector() { Id = "link23", Type =
Segments.Straight, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        connector.Add(new DiagramConnector() { Id = "link33", Type =
Segments.Bezier, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        ViewBag.Connectors = connector;
        List<SymbolPalettePalette> palettes = new
List<SymbolPalettePalette>();
        palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
= true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
= true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "connectors",
Expanded = true, Symbols = connector, IconCss = "e-ddb-icons e-connector",
Title = "Connectors" });
        ViewBag.palettes = palettes;
        ViewBag.getNodeDefaults = "getNodeDefaults";
        ViewBag.getSymbolInfo = "getSymbolInfo";
        return View();
    }
}

```

The [symbolMargin](#) property is used to create the space around elements, outside of any defined borders.

Symbol preview

The symbol preview size of the palette items can be customized using [symbolPreview](#). The [width](#) and [height](#) properties of SymbolPalette enables to define the preview size to all the symbol palette items. The [offset](#) of the dragging helper relative to the mouse cursor.

PREVIEW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
            SymbolPalettee.Add(new FlowShapes() { Id = "Terminator", Shape =
new { type = "Flow", shape = "Terminator" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Terminator", Shape =
new { type = "Flow", shape = "Terminator" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Process", Shape = new
{ type = "Flow", shape = "Process" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Decision", Shape =
new { type = "Flow", shape = "Decision" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Document", Shape =
new { type = "Flow", shape = "Document" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "PreDefinedProcess",
Shape = new { type = "Flow", shape = "PreDefinedProcess" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "PaperTap", Shape =
new { type = "Flow", shape = "PaperTap" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "DirectData", Shape =
new { type = "Flow", shape = "DirectData" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "SequentialData",
Shape = new { type = "Flow", shape = "SequentialData" } });
            List<DiagramNode> BasicNodes = new List<DiagramNode>();
            BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Rectangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Ellipse } });
            BasicNodes.Add(new DiagramNode() { Id = "Parallelogram", Shape =
new DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Parallelogram } });
            BasicNodes.Add(new DiagramNode() { Id = "Triangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Triangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Hexagon } });
```

```

        BasicNodes.Add(new DiagramNode() { Id = "Pentagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Pentagon } });
        BasicNodes.Add(new DiagramNode() { Id = "Cylinder", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Cylinder } });
        BasicNodes.Add(new DiagramNode() { Id = "Star", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Star } });
        ViewBag.BasicShapes = BasicNodes;
        List<DiagramConnector> connector = new List<DiagramConnector>();
        connector.Add(new DiagramConnector() { Id = "link1", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.Arrow },
Style = new DiagramStrokeStyle() { StrokeWidth = 2 } });
        connector.Add(new DiagramConnector() { Id = "link3", Type =
Segments.Orthogonal, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        connector.Add(new DiagramConnector() { Id = "Link21", Type =
Segments.Straight, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.Arrow },
Style = new DiagramStrokeStyle() { StrokeWidth = 2 } });
        connector.Add(new DiagramConnector() { Id = "link23", Type =
Segments.Straight, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        connector.Add(new DiagramConnector() { Id = "link33", Type =
Segments.Bezier, SourcePoint = new DiagramPoint() { X = 0, Y = 0 },
TargetPoint = new DiagramPoint() { X = 40, Y = 40 }, TargetDecorator = new
ConnectorTargetDecoratorConnectors() { Shape = DecoratorShapes.None } });
        ViewBag.Connectors = connector;
        List<SymbolPalettePalette> palettes = new
List<SymbolPalettePalette>();
        palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
= true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
= true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "connectors",
Expanded = true, Symbols = connector, IconCss = "e-ddb-icons e-connector",
Title = "Connectors" });
        ViewBag.palettes = palettes;
        ViewBag.getNodeDefaults = "getNodeDefaults";
        ViewBag.getSymbolInfo = "getSymbolInfo";
        return View();
    }
}

```

Default settings

While adding more number of symbols such as nodes and connectors to the palette, define the default settings for those objects through the [getNodeDefaults](#) and the [getConnectorDefaults](#) properties of diagram allows to define the default settings for nodes and connectors.

CSHTML

```
@(Html.EJS().SymbolPalette("symbolPalette").Palettes(ViewBag.Palette).GetSymbolInfo("getSymbolInfo").Width("100%).Height("570px").SymbolHeight(48).SymbolWidth(48).Render())
```

DEFAULT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
            SymbolPalettee.Add(new FlowShapes() { Id = "Process", Shape = new
            { type = "Flow", shape = "Process" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Document", Shape =
            new { type = "Flow", shape = "Document" } });
            List<DiagramNode> BasicNodes = new List<DiagramNode>();
            BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
            DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
            BasicShapes.Rectangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
            DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
            BasicShapes.Ellipse } });
            BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
            DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
            BasicShapes.Hexagon } });
            ViewBag.BasicShapes = BasicNodes;
            List<SymbolPalettePalette> palettes = new
            List<SymbolPalettePalette>();
            palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
            = true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
            "Flow Shapes" });
            palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
            = true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
            Shapes" });
            ViewBag.palettes = palettes;
            ViewBag.getSymbolInfo = "getSymbolInfo";
            return View();
        }
    }
}
```

```
`javascript
function getNodeDefaults(symbol) {
    if (symbol.id === 'Terminator' || symbol.id === 'Process') {
        symbol.width = 80;
        symbol.height = 40;
    } else if (symbol.id === 'Document' || symbol.id === 'PreDefinedProcess' ||
        symbol.id === 'PaperTap' || symbol.id === 'DirectData') {
        symbol.width = 50;
        symbol.height = 40;
    }
    symbol.style = {
        strokeWidth: 2
    };
}

function getSymbolInfo(symbol) {
    return {
        fit: true
    };
}
```

Adding symbol description for symbols in the palette

The diagram provides support to add symbol description below each symbol of a palette. This descriptive representation of each symbol will enhance the details of the symbol visually. The height and width of the symbol description can also be set individually.

- The property `getSymbolInfo` can be used to add the symbol description at runtime.

DESCRIPTION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
        }
    }
}
```

```

        SymbolPallettee.Add(new FlowShapes() { Id = "Process", Shape = new
        { type = "Flow", shape = "Process" } });
        SymbolPallettee.Add(new FlowShapes() { Id = "Document", Shape =
        new { type = "Flow", shape = "Document" } });
        List<DiagramNode> BasicNodes = new List<DiagramNode>();
        BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
        DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
        BasicShapes.Rectangle } });
        BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
        DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
        BasicShapes.Ellipse } });
        BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
        DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
        BasicShapes.Hexagon } });
        ViewBag.BasicShapes = BasicNodes;
        List<SymbolPalettePalette> palettes = new
        List<SymbolPalettePalette>();
        palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
        = true, Symbols = SymbolPallettee, IconCss = "e-ddb-icons e-basic", Title =
        "Flow Shapes" });
        palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
        = true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
        Shapes" });
        ViewBag.palettes = palettes;
        ViewBag.getSymbolInfo = "getSymbolInfo";
        return View();
    }
}

```

```
`javascript
```

```

function getSymbolInfo(symbol) {
    if (symbol['text'] !== undefined) {
        return {
            width: 75,
            height: 40,
            //Add or Remove the Text for Symbol palette item.
            description: {
                //Defines the symbol description
                text: symbol['text'],
                //Defines how to handle the text when its size exceeds the given symbol size
                overflow: 'Wrap'
            }
        };
    }
}

```

```
return {  
  width: 75,  
  height: 40,  
  description: {  
    text: symbol.shape['shape']  
  }  
};  
}
```

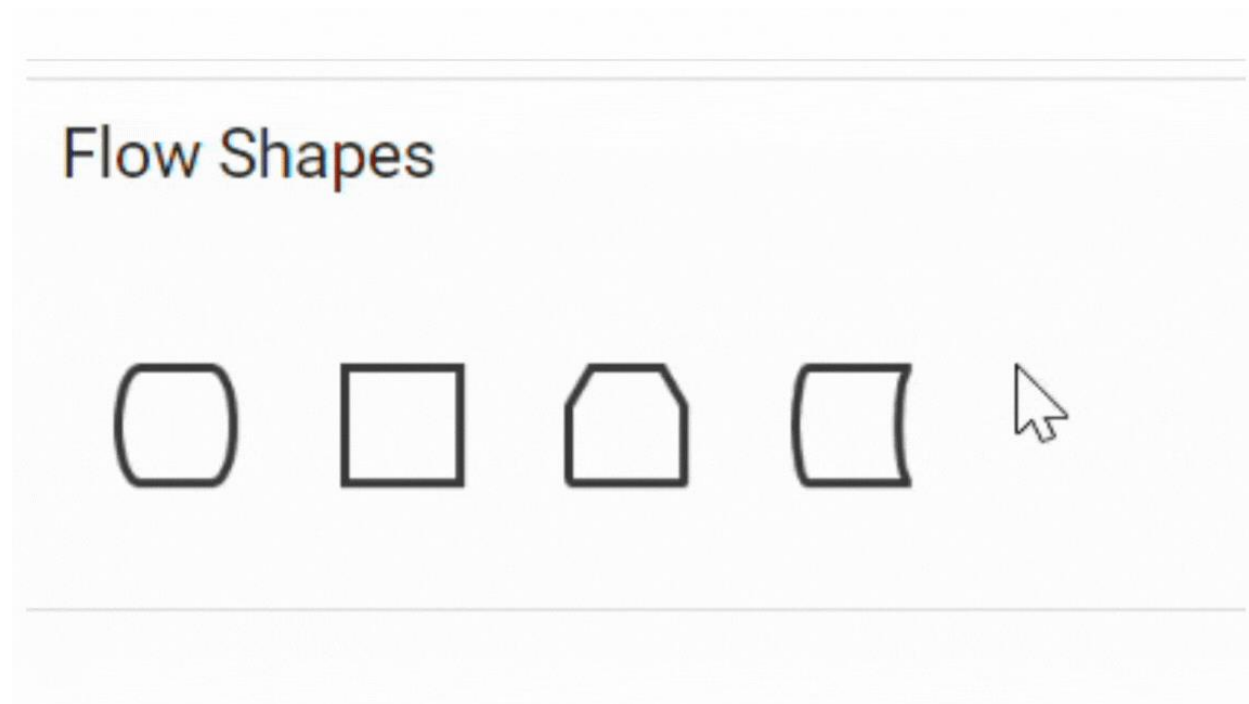
Tooltip for symbols in symbol palette

The Symbol palette supports displaying tooltips when mouse hovers over the symbols. You can customize the tooltip for each symbol in the symbol palette.

Default tooltip for symbols

When hovering over symbols in the symbol palette, the default tooltip displays the symbol's ID.

Refer to the image below for an illustration of the tooltip behavior in the symbol palette.



Custom tooltip for symbols

To customize the tooltips for symbols in the symbol palette, assign a custom tooltip to the 'Tooltip' content property of each symbol. Once you define the custom tooltip, enable the Tooltip constraints for each symbol, ensuring that the tooltips are displayed when users hover over them.

Here, the code provided below demonstrates how to define tooltip content to symbols within a symbol palette.

DESCRIPTION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using System.Drawing;
namespace EJ2MVCSampleBrowser.Controllers.Diagram {
    public partial class DiagramController: Controller {
        // GET: Nodes
        public ActionResult Nodes() {
            List<DiagramNode> SymbolPalettee = new List<DiagramNode>();
            SymbolPalettee.Add(new FlowShapes() { Id = "Process", Shape = new
{ type = "Flow", shape = "Process" } });
            SymbolPalettee.Add(new FlowShapes() { Id = "Document", Shape =
new { type = "Flow", shape = "Document" } });
            List<DiagramNode> BasicNodes = new List<DiagramNode>();
            BasicNodes.Add(new DiagramNode() { Id = "Rectangle", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Rectangle } });
            BasicNodes.Add(new DiagramNode() { Id = "Ellipse", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Ellipse },
            //To display customized tooltip content by enabling Tooltip
Constraints
            Tooltip = new DiagramDiagramTooltip() { Content = "Customised
Tooltip"},
            Constraints = NodeConstraints.Default | NodeConstraints.Tooltip
});
            BasicNodes.Add(new DiagramNode() { Id = "Hexagon", Shape = new
DiagramBasicShape() { Type = Syncfusion.EJ2.Diagrams.Shapes.Basic, Shape =
BasicShapes.Hexagon } });
            ViewBag.BasicShapes = BasicNodes;
            List<SymbolPalettePalette> palettes = new
List<SymbolPalettePalette>();
            palettes.Add(new SymbolPalettePalette() { Id = "flow", Expanded
= true, Symbols = SymbolPalettee, IconCss = "e-ddb-icons e-basic", Title =
"Flow Shapes" });
            palettes.Add(new SymbolPalettePalette() { Id = "basic", Expanded
= true, Symbols = BasicNodes, IconCss = "e-ddb-icons e-flow", Title = "Basic
Shapes" });
            ViewBag.palettes = palettes;
            ViewBag.getSymbolInfo = "getSymbolInfo";
            return View();
        }
    }
}

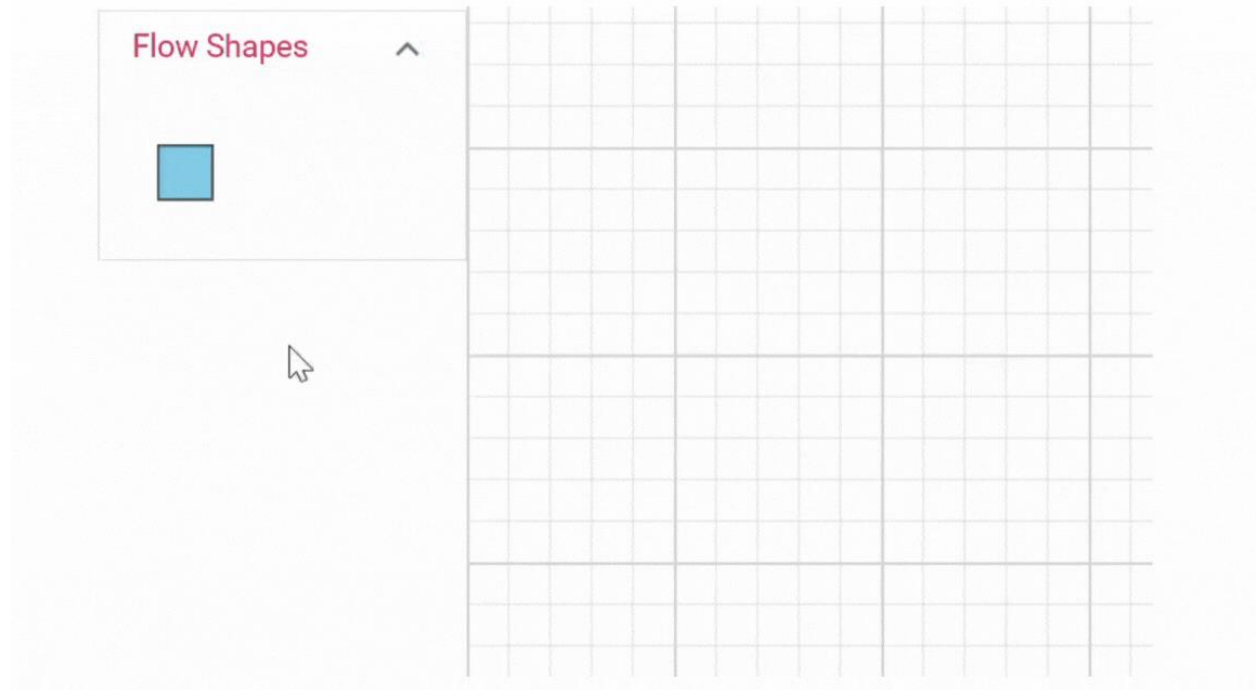
```

How to provide different tooltip for Symbol palette and diagram elements.

Differentiate the tooltips between symbols in the symbol palette and dropped nodes by utilizing the `dragEnter` event. When a custom tooltip is defined for a symbol, it will be displayed for both the symbol and the dropped node in the diagram canvas. However, to provide distinct tooltips for symbols in the palette and dropped nodes, capture the `dragEnter` event and assign specific tooltips dynamically.

When a symbol is dragged from the symbol palette and enters the diagram canvas, the `[DragEnter]` `IDragEnterEventArgs` event is triggered. Within this event, you can define a new tooltip for the dropped node. By assigning custom tooltip content to the `Tooltip` property of the node, you can provide a distinct tooltip that is specific to the dropped node.

The following image illustrates the differentiation of tooltips displayed in the Symbol Palette and the Diagram.



The following code snippet will demonstrate how to define two different tooltip for symbol in the symbol palette and dropped node in the diagram canvas.

```
`js
//Initialize the Diagram
let diagram: Diagram = new Diagram({
width: '100%', height: '500px',
connectors: connectors, nodes: nodes,
//event to change tooltip content while dragging symbols into Diagram
dragEnter: dragEnter,
});
diagram.appendTo('#diagram');
function dragEnter(args:IDragEnterEventArgs)
{
//enable tooltip connstraints for the dragged symbol
args.dragItem.constraints = NodeConstraints.Default | NodeConstraints.Tooltip;
```

```
//change the tooltip content of the dragged symbol  
args.dragItem.tooltip.content='This is Diagram Tooltip';  
}  
,
```

Palette interaction

Palette interaction notifies the element enter, leave, and dragging of the symbols into the diagram.

DragEnter

[DragEnter] `IDragEnterEventArgs` notifies, when the element enters into the diagram from symbol palette.

DragLeave

[DragLeave] `IDragLeaveEventArgs` notifies, when the element leaves from the diagram.

DragOver

[DragOver] `IDragOverEventArgs` notifies, when an element is dragged over another diagram element.

Note: The diagram provides support to cancel the drag and drop operation from the symbol palette to the diagram when the ESC key is pressed.

See Also

- [How to add the symbol to the diagram](#)

Overview Control

Overview control allows to see a preview or an overall view of the entire content of a diagram. This helps you to look at the overall picture of a large diagram and also to navigate, pan, or zoom, on a particular position of the page.

When you work on a very large diagram, you may not know the part you are actually working on, or navigation from one part to another might be difficult. One solution for navigation is to zoom out the entire diagram and find where you are. Then, you can zoom in a particular area you want to. This solution is not suitable when you need some frequent navigation.

Overview control solves these problems by showing a preview, that is, an overall view of the entire diagram. A rectangle indicates viewport of the diagram. Navigation becomes easy by dragging this rectangle.

Create overview

The `sourceID` property of overview should be set with the corresponding diagram ID for the overall view.

The `width` and `height` properties of the overview allows to define the size of the overview.

Zoom and Pan

In overview, the view port of the diagram is highlighted with a red colored rectangle. Diagram can be zoomed/panned by interacting with that. You can interact with overview as follows:

- Resize the rectangle: Zooms in/out the diagram.
- Drag the rectangle: Pans the diagram.

- Click at a position: Navigates to the clicked region.
- Choose a particular region by clicking and dragging: Navigates to the specified region.

CSHTML

```
<div class="col-lg-12 control-section">
    <div class="content-wrapper">

@Html.EJS().Diagram("diagram").Width("100%").Height("590px").GetNodeDefaults(
    "getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").ScrollSettings(s =>
s.ScrollLimit(Syncfusion.EJ2.Diagrams.ScrollLimit.Infinity)).DataSourceSettings(ss => ss.Id("Id").ParentId("ReportingPerson")
    .DataManager(new DataManager() { Data =
    (List<OverviewData>)ViewBag.Nodes })).Layout(l =>
l.Type(Syncfusion.EJ2.Diagrams.LayoutType.OrganizationalChart).GetLayoutInfo("getLayoutInfo").SnapSettings(se =>
se.Constraints(Syncfusion.EJ2.Diagrams.SnapConstraints.None)).SetNodeTemplate("setNodeTemplate").Mode(Syncfusion.EJ2.Diagrams.RenderingMode.SVG).Created(
    "diagramCreated").Render()
    </div>
</div>
<div class="col-lg-4 property-section"
style="padding:0px;right:49px;bottom:38px;border: #eeeeee;border-style:
solid;box-shadow: 0px 2px 2px rgba(0,0,0,0.3);
background:#f7f7f7;position:absolute">

@Html.EJS().Overview("overview").Width("100%").Height("150px").SourceID("diagram").Render()
</div>
```

OVERVIEW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Diagrams;
using sample1.Models;
namespace sample1.Controllers
{
    public partial class DiagramController : Controller
    {
        // GET: Overview
        public ActionResult Overview()
        {
            ViewBag.Nodes = OverviewData.GetData();
            return View();
        }
    }
}

public class OverviewData
{
    public string Id { get; set; }
    public string Name { get; set; }
```

```

public string Designation { get; set; }
public string ReportingPerson { get; set; }
public string Image { get; set; }
public OverviewData(string id, string name, string designation,
string reportingperson)
{
    this.Id = id;
    this.Name =name;
    this.Designation =designation;
    this.ReportingPerson =reportingperson;
}
public static List<OverviewData> GetData()
{
    List<OverviewData> data = new List<OverviewData>();
    data.Add(new OverviewData("parent", "Maria Anders", "Managing
Director", ""));
    data.Add(new OverviewData("1", "Ana Trujillo", "Project
Manager", "parent"));
    data.Add(new OverviewData("2", "Anto Moreno", "Project Manager",
"parent"));
    data.Add(new OverviewData("3", "Thomas Hardy", "Project Lead",
"1"));
    data.Add(new OverviewData("4", "Christina kaff", "Project Lead",
"2"));
    data.Add(new OverviewData("5", "Hanna Moos", "Senior S/w Engg",
"3"));
    data.Add(new OverviewData("6", "Peter Citeaux", "Senior S/w
Engg", "4"));
    return data;
}
}

```

Migration from Essential JS 1

This article describes the API migration process of Diagram component from Essential JS 1 to Essential JS 2.

Background

behavior	API in Essential JS 1	API in Essential JS 2
Defines the background color of diagram elements	Property: BackgroundColor @{ Html.EJ().Diagram("diagram").BackgroundColor("yellow").Render(); }	Property: BackgroundColor @Html.EJS().Diagram("container").BackgroundColor("red").Render()
Defines how to align the background image over the	Property: BackgroundImage.Alignment @{ Html.EJ().Diagram("diagram").BackgroundImage(new BackgroundImage() { Alignment=ImageAlignment.XMidYMid}).Render(); }	Property: Background.Align @Html.EJS().Diagram("container").PageSettings(new DiagramPageSettings() { Background=new DiagramBackground() { Align=ImageAlignment.XMidYMid} }).Render()

diagram area		
Defines how the background image should be scaled/stretched	Property: BackgroundImage.Scale @ Html.EJ().Diagram("diagram").BackgroundImage(new BackgroundImage() { Scale=ScaleConstraints.Meet }).Render(); }	Property: Background.Scale Html.EJS().Diagram("container").PageSettings(new DiagramPageSettings() { Background=new DiagramBackground() { Scale=Scale.Meet } }).Render()
Sets the source path of the background image	Property: BackgroundImage.Source @ Html.EJ().Diagram("diagram").BackgroundImage(new BackgroundImage() { Source=" ../images/Employee/artBoard 13.Png" }).Render(); }	Property: Background.Source @Html.EJS().Diagram("container").PageSettings(new DiagramPageSettings() { Background=new DiagramBackground() { Source="Syncfusion.Png" } }).Render()

Bridging

behavior	API in Essential JS 1	API in Essential JS 2
Sets the direction of line bridges	Property: BridgeDirection @ Html.EJ().Diagram("diagram").BridgeDirection(BridgeDirection.Bottom).Render(); }	Property: BridgeDirection @Html.EJS().Diagram("container").BridgeDirection(BridgeDirection.Left).Render()

CommandManager

behavior	API in Essential JS 1	API in Essential JS 2
Stores the multiple commands with the corresponding command	Property: CommandManager.Commands [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties); function canExecute(args) { var diagram = \$("#DiagramContent").EjDiagram("instance"); return diagram.Model.SelectedItems.Children.Length; } function execute(args) {	Property: CommandManager.Commands [View] @Html.EJS().Diagram("container").CommandManager(ViewBag.commandManager).Render()[Model] List commands = new List(); commands.Add(new DiagramCommand() { Name = "customCopy" }); DiagramCommandManager commandManager = new DiagramCommandManager() { Commands = commands }; ViewBag.commandManager = commandManager;

nd objects	<pre> var diagram = \$("#DiagramContent").EjDiagram("ins tance"); diagram.Copy(); diagram.Paste();][Model] DiagramProperties Model = new DiagramProperties(); Command clone = new Command() { Execute = "executeClone", CanExecute = "canExecuteClone", Gesture = new Gesture() { Key = Keys.C, KeyModifiers = KeyModifiers.Shift } }; Model.CommandManager.Commands .Add("clone", clone); ViewData["diagramModel"] = Model; </pre>	
Defines any additio nal parame ters that are require d at runtime	<pre> Property:CommandManager.Comm ands.Parameter[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as Syncfusion.JavaScript.DataVisualizatio n.Models.DiagramProperties); function canExecute(args) { var diagram = \$("#DiagramContent").EjDiagram("ins tance"); return diagram.Model.SelectedItems.Childre n.Length; } function execute(args) { var diagram = \$("#DiagramContent").EjDiagram("ins tance"); diagram.Copy(); diagram.Paste();][Model] DiagramProperties Model = new DiagramProperties(); Command clone = new Command() { parameter : "node", Execute = "executeClone", CanExecute = "canExecuteClone", Gesture = new Gesture() { Key = Keys.C, KeyModifiers = KeyModifiers.Shift } }; Model.CommandManager.Commands .Add("clone", clone); ViewData["diagramModel"] = Model; </pre>	<pre> Property:CommandManager.Commands.Parameter [View] @Html.EJS().Diagram("container").CommandManager(Vi ewBag.commandManager).Render()[Model] List commands = new List(); commands.Add(new DiagramCommand() { Parameter = "node" }); DiagramCommandManager commandManager = new DiagramCommandManager() { Commands = commands }; ViewBag.commandManager = commandManager; </pre>

Connectors

behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Allows the user to save custom information/data about a connector	Property: Connector.AddInfo @ Dictionary AddInfo = new Dictionary(); AddInfo.Add("Description", "Bidirectional Flow"); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { AddInfo = AddInfo })).Render(); }	Property: Connectors.AddInfo [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] Dictionary AddInfo = new Dictionary(); AddInfo.Add("Description", "Bidirectional Flow"); List connectors = new List(); connectors.Add(new DiagramConnector() { AddInfo = AddInfo }); ViewBag.Connectors = connectors;
Defines the bridge space of connector	Property: Connector.BridgeSpace @ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { BridgeSpace = 15 })).Render(); }	Property: Connectors.BridgeSpace [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { BridgeSpace = 15 }); ViewBag.Connectors = connectors;
Enables or disables the behaviors of connectors	Property: Connector.Constraints @ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Constraints = Syncfusion.JavaScript.DataVisualization.DiagramEnums.ConnectorConstraints.Bridging })).Render(); }	Property: Connector.Constraints [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Constraints = ConnectorConstraints.Bridging }); ViewBag.Connectors = connectors;
Defines the radius of the rounded corner	Property: Connector.CnerRadius @ Collection Segment = new Collection(); Segment.Add(new Segment() { Type = Segments.Orthogonal }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { CornerRadius = 15 })).Render(); }	Property: Connector.CnerRadius [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { CornerRadius = 5 }); ViewBag.Connectors = connectors;
Customize connectors appearance using user-	Property: Connector.CssClass //CSS style .HoverConnector:hover { stroke:blue } @ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() {	Not applicable

defined CSS	<pre>CssClass = "hoverConnector" })).Render(); }</pre>	
Sets the horizontal alignment of the connector	<pre>Property:Connector.HorizontalAlign @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { HorizontalAlign = HorizontalAlignment.Right })).Render(); }</pre>	Not applicable
A collection of JSON objects where each object represents a label	<pre>Property:Connector.Labels @{ Collection Labels = new Collection(); Labels.Add(new Label() { Text = "Connector" }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Labels = Labels })).Render(); }</pre>	<pre>Property:Connector.Annotations [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List annotation = new List(); annotation.Add(new DiagramConnectorAnnotation() { Content = "connector" }); List connectors = new List(); connectors.Add(new DiagramConnector() { Annotations = annotation}); ViewBag.Connectors = connectors;</pre>
Stroke color of the connector	<pre>Property:Connector.LineColor @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Name = "connector1", LineColor = "blue" })).Render(); }</pre>	<pre>Property:Connector.Style.StrokeColor [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Style = { StrokeColor = "blue" } }); ViewBag.Connectors = connectors;</pre>
Sets the pattern of dashes and gaps used to stroke the path of the connector	<pre>Property:Connector.LineDashArray @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { LineDashArray = "2,2" })).Render(); }</pre>	<pre>Property:Connector.Style.StrokeDashArray [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Style = { StrokeDashArray= "2, 2" } }); ViewBag.Connectors = connectors;</pre>
Sets the width	<pre>Property:Connector.LineWidth</pre>	<pre>Property:Connector.Style.StrokeWidth</pre>

of the line	<pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { LineWidth = 10 })).Render(); }</pre>	<pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Style = { StrokeWidth = 2 }}); ViewBag.Connectors = connectors;</pre>
Defines the padding value to ease the interaction with connectors	<p>Property: Connector.LineHitPadding</p> <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { LineHitPadding = 15 })).Render(); }</pre>	<p>Property: Connectors.HitPadding</p> <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { HitPadding = 20 }); ViewBag.Connectors = connectors;</pre>
Sets a unique name for the connector	<p>Property: Connector.Name</p> <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Name = "connector1" })).Render(); }</pre>	<p>Property: Connectors.Id</p> <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Id = "connector1" }); ViewBag.Connectors = connectors;</pre>
Defines the transparency of the connector	<p>Property: Connector.Opacity</p> <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Opacity = 1 })).Render(); }</pre>	<p>Property: Connectors.Style.Opacity</p> <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Style = { Opacity = 0.5 }}); ViewBag.Connectors = connectors;</pre>
Sets the parent name of the connector.	<p>Property: Connector.Parent</p> <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Parent = "parent" })).Render(); }</pre>	Not applicable
An array of JSON objects where each object	<p>Property: Connector.Segments</p> <pre>@{ Collection Segment = new Collection(); Segment.Add(new Segment() { Type = Segments.Orthogonal }); Html.EJ().Diagram("diagram").Connectors</pre>	<p>Property: Connector.Segments</p> <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List segments = new List(); segments.Add(new Segment() { Type = Segments.Orthogonal }); List connectors = new List();</pre>

represents a segment	rs(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }	connectors.Add(new DiagramConnector() { Segments = segments }); ViewBag.Connectors = connectors;
Sets the direction of orthogonal segment	Property:Connector.Segments.Direction @{ Collection Segment = new Collection(); Segment.Add(new Segment() { Direction = "bottom" }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }	Property:Connectors.Segments.Direction [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List segments = new List(); segments.Add(new Segment() { Direction = "Bottom" }); List connectors = new List(); connectors.Add(new DiagramConnector() { Segments = segments }); ViewBag.Connectors = connectors;
Describes the length of orthogonal segment	Property:Connector.Segments.Length @{ Collection Segment = new Collection(); Segment.Add(new Segment() { Length = 50 }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }	Property:Connectors.Segments.Length [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List segments = new List(); segments.Add(new Segment() { Length=30 }); List connectors = new List(); connectors.Add(new DiagramConnector() { Segments = segments }); ViewBag.Connectors = connectors;
Describes the end point of bezier/straight segment	Property:Connector.Segments.Point @{ Collection Segment = new Collection(); Segment.Add(new Segment() { Point = new DiagramPoint(75, 150) }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }	Property:Connectors.Segments.Point
Defines the first control point of the bezier segment	Property:Connector.Segments.Point1 @{ Collection Segment = new Collection(); Segment.Add(new Segment() { Point1 = new DiagramPoint(75, 150) }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }	Property:Connectors.Segments.Point1
Defines the second control	Property:Connector.Segments.Point2 @{ Collection Segment = new Collection(); Segment.Add(new	Property:Connectors.Segments.Point2

point of bezier segment	<pre>Segment() { Point2 = new DiagramPoint(75, 150) }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }</pre>	
Sets the type of the segment	<pre>@{ Collection Segment = new Collection(); Segment.Add(new Segment() { Type = Segments.Straight }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }</pre>	Property: Connectors.Segments.Type [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List segments = new List(); segments.Add(new Segment() { Type = Segments.Bezier }); List connectors = new List(); connectors.Add(new DiagramConnector() { Segments = segments }); ViewBag.Connectors = connectors;
Describes the length and angle between in the first control point and the start point of bezier segment	Property: Connector.Segments.Vector1 <pre>@{ Collection Segment = new Collection(); Segment.Add(new Segment() { Vector1 = new Vector(75, 0) }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }</pre>	Property: Connectors.Segments.Vector1
Describes the length and angle between in the second control point and end point of bezier segment	Property: Connector.Segments.Vector2 <pre>@{ Collection Segment = new Collection(); Segment.Add(new Segment() { Vector2 = new Vector(75, 180) }); Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ Segments=Segment })) .Render(); }</pre>	Property: Connectors.Segments.Vector2

Sets the type of the connector	Property: Connector.Shape.Type @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Shape = new Shape() { Type = Shapes.BPMN } })).Render(); }	Property: Connectors.Shape.Type [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Shape= new BpmnConnectors() { Type = "Bpmn" } }); ViewBag.Connectors = connectors;
Defines the source decorator of the connector	Property: Connector.SourceDecorator @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourceDecorator = new Decorator() { Shape=DecoratorShapes.OpenArrow } })).Render(); }	Property: Connectors.SourceDecorator [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Shape = DecoratorShapes.OpenArrow } } }); ViewBag.Connectors = connectors;
Sets the border color of the source decorator	Property: Connector.SourceDecorator.BorderColor @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourceDecorator = new Decorator() { BorderColor = "red" } })).Render(); }	Property: Connectors.SourceDecorator.Style.StrokeColor [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Style = { StrokeColor="green" } } }); ViewBag.Connectors = connectors;
Sets the border width of the source decorator	Property: Connector.SourceDecorator.BorderWidth @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourceDecorator = new Decorator() { BorderWidth = 5 } })).Render(); }	Property: Connectors.SourceDecorator.Style.StrokeWidth [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Style = { StrokeWidth=2 } } }); ViewBag.Connectors = connectors;
Defines to customize source decorator appear	Property: Connector.SourceDecorator.CssClass @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() {	Not applicable

ance using user- defined CSS	SourceDecorator = new Decorator() { CssClass = "className" } }).Render(); }	
Sets the fill color of the source decorat or	Property:Connector.SourceDecorator.FillColor @{ Html.EJ().Diagram("diagram").Height("500px").Width("500px").Connectors(c => c.Add(new Connector(){SourceDecorator = new Decorator() { FillColor = "red" } })).Render(); }	Property:Connectors.SourceDecorator.Style.Fill [View] @Html.EJS().Diagram("container").Width("1000").Height("645px").Connectors(ViewBag.Connectors).Render() [Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Style = { Fill= "red" } } }); ViewBag.Connectors = connectors;
Sets the height of the source decorat or	Property:Connector.SourceDecorator.Height @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ SourceDecorator = new Decorator() { Height = 10 } })).Render(); }	Property:Connectors.SourceDecorator.Height [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render() [Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Height=10 } }); ViewBag.Connectors = connectors;
Defines the custom shape of the source decorat or	Property:Connector.SourceDecorator.PathData @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ SourceDecorator = new Decorator() { Shape = DecoratorShapes.Path, PathData = "M 376.892,225.284L 371.279,211.95L 376.892,198.617L 350.225,211.95L 376.892,225.284 Z" } })).Render(); }	Property:Connectors.SourceDecorator.PathData [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render() [Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Shape = DecoratorShapes.Custom, PathData= "M 376.892,225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" } }); ViewBag.Connectors = connectors;
Defines the shape of the source decorat or.	Property:Connector.SourceDecorator.Shape @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ SourceDecorator = new Decorator() { Shape = OpenArrow } })).Render(); }	Property:Connectors.SourceDecorator.Shape [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render() [Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Shape = DecoratorShapes.OpenArrow } }); ViewBag.Connectors = connectors;

Defines the width of the source decorator	Property: Connector.SourceDecorator.Width <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourceDecorator = new Decorator() { Width = 10 } })).Render(); }</pre>	Property: Connectors.SourceDecorator.Width [View] <pre>@Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceDecorator = new DiagramDecorator () { Width=10 } }); ViewBag.Connectors = connectors;</pre>
Sets the source node of the connector	Property: Connector.SourceNode <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourceNode = "source" })).Render(); }</pre>	Property: Connectors.SourceID [View] <pre>@Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourceID="source" } }); ViewBag.Connectors = connectors;</pre>
Defines the space to be left between the source node and the source point of a connector	Property: Connector.SourcePadding <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourcePadding = 2 })).Render(); }</pre>	Property: Connectors.HitPadding [View] <pre>@Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { HitPadding=2 } }); ViewBag.Connectors = connectors;</pre>
Describes the start point of the connector	Property: Connector.SourcePoint <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { SourcePoint = new DiagramPoint(100, 100) })).Render(); }</pre>	Property: Connectors.SourcePoint [View] <pre>@Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { SourcePoint = { X= 100, Y=100 } }); ViewBag.Connectors = connectors;</pre>
Sets the source port of the	Property: Connector.SourcePort <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() {</pre>	Property: Connectors.SourcePortID [View] <pre>@Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() {</pre>

connect or	SourcePort = "sourcePort" })).Render(); }	SourceID="source", SourcePortID="sourcePortId" }); ViewBag.Connectors = connectors;
Defines the target decorat or of the connect or	Property:Connector.TargetDecorator @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { Shape=DecoratorShapes.OpenArrow } })).Render(); }	Property:Connectors.TargetDecorator [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Shape = DecoratorShapes.OpenArrow } }); ViewBag.Connectors = connectors;
Sets the border color of the target decorat or	Property:Connector.TargetDecorator.BorderColor @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { BorderColor = "red" } })).Render(); }	Property:Connectors.TargetDecorator.Style.StrokeColor [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Style = { StrokeColor="green" } } }); ViewBag.Connectors = connectors;
Sets the border width of the decorat or	Property:Connector.TargetDecorator.BorderWidth @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { Shape=DecoratorShapes.OpenArrow, BorderWidth = 5 } })).Render(); }	Property:Connectors.TargetDecorator.Style.StrokeWidth [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Style = { StrokeWidth=2 } } }); ViewBag.Connectors = connectors;
Defines to customi ze target Decorat or appear ance using user- defined CSS	Property:Connector.TargetDecorator.CssClass @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { CssClass = "className" } })).Render(); }	Not applicable

Sets the fill color of the target decorator	Property: Connector.TargetDecorator.FillColor <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { FillColor = "red" } })).Render(); }</pre>	Property: Connectors.TargetDecorator.Style.Fill <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Style = {Fill= "red" } } }); ViewBag.Connectors = connectors;</pre>
Sets the height of the target decorator	Property: Connector.TargetDecorator.Height <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { Height = 10 } })).Render(); }</pre>	Property: Connectors.TargetDecorator.Height <pre>[View] @Html.EJS().Diagram("container").Width("1000").Height("645px").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Height=10 } }); ViewBag.Connectors = connectors;</pre>
Defines the custom shape of the target decorator	Property: Connector.TargetDecorator.PathData <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { shape: DecoratorShapes.Path, pathData: "M 376.892,225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" } })).Render(); }</pre>	Property: Connectors.TargetDecorator.PathData <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Shape = DecoratorShapes.Custom, PathData= "M 376.892,225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" } }); ViewBag.Connectors = connectors;</pre>
Defines the shape of the target decorator.	Property: Connector.TargetDecorator.Shape <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { Shape = DecoratorShapes.OpenArrow } })).Render(); }</pre>	Property: Connectors.TargetDecorator.Shape <pre>[View] @Html.EJS().Diagram("container").Width("1000").Height("645px").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Shape = DecoratorShapes.OpenArrow } }); ViewBag.Connectors = connectors;</pre>
Defines the width of the target	Property: Connector.TargetDecorator.Width <pre>@{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { Width = 1000 } })).Render(); }</pre>	Property: Connectors.TargetDecorator.Width <pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Width=1000 } }); ViewBag.Connectors = connectors;</pre>

decorator	rs(c => c.Add(new Connector(){ TargetDecorator = new Decorator() { Width = 10 } })).Render(); }	List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { Width=10 } }); ViewBag.Connectors = connectors;
Sets the target node of the connector	Property: Connector.TargetNode @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetNode = "target" })).Render(); }	Property: Connectors.TargetID [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetID="target" }); ViewBag.Connectors = connectors;
Defines the space to be left between the target node and the target point of a connector	Property: Connector.TargetPadding @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetPadding = 2 })).Render(); }	Property: Connectors.HitPadding [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetDecorator = new DiagramDecorator () { HitPadding = 2 }); ViewBag.Connectors = connectors;
Describes the start point of the connector	Property: Connector.TargetPoint @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){Name = "connector1", TargetPoint = new DiagramPoint(200, 200)})).Render(); }	Property: Connectors.TargetPoint [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetPoint = { X = 200, Y = 200 } } }); ViewBag.Connectors = connectors;
Sets the target port of the connector	Property: Connector.TargetPort @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector(){ TargetPort = "targetPort" })).Render(); }	Property: Connectors.TargetPortID [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { TargetID="target", TargetPortID="targetPortId" }); ViewBag.Connectors = connectors;
Defines the	Property: Connector.Tooltip	Property: Connectors.Tooltip

tooltip that should be shown when the mouse hovers over connector	<pre>@{ Html.EJ().Diagram("diagram").Height("500px").Width("500px").Connectors(c => c.Add(new Connector() { Name = "connector1", SourcePoint = new DiagramPoint(100, 100), TargetPoint = new DiagramPoint(200, 200), Segments = Segment, Tooltip = new Tooltip() { TemplateId = "mouseOverTooltip" } })).Render(); }</pre>	<pre>[View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Constraints=ConnectorConstraints.Default ConnectorConstraints.Tooltip, Tooltip = { Content = "Connector" } }); ViewBag.Connectors = connectors;</pre>
Sets the vertical alignment of connector	<pre>Property:Connector.VerticalAlign @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { VerticalAlign= VerticalAlignment.Bottom})).Render(); }</pre>	Not applicable
Enables or disables the visibility of connector	<pre>Property:Connector.Visible @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Visible=false })).Render(); }</pre>	<pre>Property:Connectors.Visible [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { Visible = true }); ViewBag.Connectors = connectors;</pre>
Sets the z-index of the connector	<pre>Property:Connector.ZOrder @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { ZOrder = 3 })).Render(); }</pre>	<pre>Property:Connectors.ZIndex [View] @Html.EJS().Diagram("container").Connectors(ViewBag.Connectors).Render()[Model] List connectors = new List(); connectors.Add(new DiagramConnector() { ZIndex = -1 }); ViewBag.Connectors = connectors;</pre>
Enables/Disables the default behaviors of the diagram	<pre>Property:Constraints @{ Html.EJ().Diagram("diagram").Connectors(c => c.Add(new Connector() { Constraints=ConnectorConstraints.Default ConnectorConstraints.Bridging })).Render(); }</pre>	<pre>Property:Constraints @Html.EJS().Diagram("container").Constraints(DiagramConstraints.Default DiagramConstraints.Bridging).Render()</pre>

ContextMenu

behavior	API in Essential JS 1	API in Essential JS 2
Defines the collection of context menu items	Property: ContextMenu.Items [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties); [Model] DiagramProperties Model = new DiagramProperties(); List<MenuItem> menuItems = new List(); menuItems.Add(new MenuItem() { Name = "HyperLink" }); Model.ContextMenu = new DiagramContextMenu() { Items = menuItems }; ViewData["diagramModel"] = Model;	Property: ContextMenuSettings.Items [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List<MenuItem> menuItems = new List(); menuItems.Add(new MenuItem() { Text = "delete" }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings;
Defines the text for the collection of context menu item	Property: ContextMenu.Items.Text [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties); [Model] DiagramProperties Model = new DiagramProperties(); List<MenuItem> menuItems = new List(); menuItems.Add(new MenuItem() { Text = "Text" }); Model.ContextMenu = new DiagramContextMenu() { Items = menuItems }; ViewData["diagramModel"] = Model;	Property: ContextMenuSettings.Items.Text [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List<MenuItem> menuItems = new List(); menuItems.Add(new MenuItem() { Text = "delete" }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings;
Defines the name for the collection of context menu items	Property: ContextMenu.Items.Name [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties); [Model] DiagramProperties Model = new DiagramProperties(); List	Property: ContextMenuSettings.Items.Id [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List<MenuItem> menuItems = new List(); menuItems.Add(new MenuItem() { Id = "id" }); DiagramContextMenuSettings contextMenuSettings = new

	<pre> menuItems = new List(); menuItems.Add(new ContextMenu() { Name = "HyperLink" }); Model.ContextMenu = new DiagramContextMenu() { Items = menuItems }; ViewData["diagramModel"] = Model; </pre>	<pre> DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings; </pre>
Defines the image url for the collection of context menu items	<pre> Property: ContextMenu.Items.ImageUrl [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); List menuItems = new List(); menuItems.Add(new ContextMenu() { ImageUrl = "Images/zoomIn.Png" }); Model.ContextMenu = new DiagramContextMenu() { Items = menuItems }; ViewData["diagramModel"] = Model; </pre>	<pre> Property: ContextMenuSettings.Items.Url [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List menuItems = new List(); menuItems.Add(new ContextMenu() { Url = "Images/zoomIn.Png" }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings; </pre>
Defines the cssClass for the collection of context menu items	<pre> Property: ContextMenu.Items.CssClass [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); List menuItems = new List(); menuItems.Add(new ContextMenu() { CssClass = "menu" }); Model.ContextMenu = new DiagramContextMenu() { Items = menuItems }; ViewData["diagramModel"] = Model; </pre>	<pre> Property: ContextMenuSettings.Items.IconCss [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List menuItems = new List(); menuItems.Add(new ContextMenu() { IconCss = "e-copy" }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings; </pre>

Defines the collection of sub items for the context menu items	Property: ContextMenu.Items.SubItems [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); List SubItems = new List(); SubItems.Add(new ContextMenuItem() { Name = "HyperLink" }); List menuItems = new List(); menuItems.Add(new ContextMenuItem() { SubItems = SubItems }); Model.ContextMenu = new DiagramContextMenu() { Items = menuItems }; ViewData["diagramModel"] = Model;	Property: ContextMenuSettings.Items.Items [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List SubItems = new List(); SubItems.Add(new ContextMenuItem() { Id = "id" }); List menuItems = new List(); menuItems.Add(new ContextMenuItem() { Items = SubItems }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings;
Set whether to display the default context menu items or not	Property: ContextMenu.ShowCustomMenuItemsOnly [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.ContextMenu = new DiagramContextMenu() { ShowCustomMenuItemsOnly = true }; ViewData["diagramModel"] = Model;	Property: ContextMenuSettings.ShowCustomMenuOnly [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] LDiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { ShowCustomMenuOnly = false }; ViewBag.contextMenuSettings = contextMenuSettings;
Specifies separator between the menu items	Not applicable	Property: ContextMenuSettings.Items.Separator [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List menuItems = new List(); menuItems.Add(new ContextMenuItem() { Separator = true }); DiagramContextMenuSettings contextMenuSettings = new

		DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings;
Define the target to show the menu item.	Not applicable	Property: ContextMenuSettings.Items.Target [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List menuItems = new List(); menuItems.Add(new ContextMenuItem() { Target = ".E-diagramcontent" }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Items = menuItems }; ViewBag.contextMenuSettings = contextMenuSettings;
Enables/Disables the context menu items	Not applicable	Property: ContextMenuSettings.Show [View] @Html.EJS().Diagram("container").ContextMenuSettings(ViewBag.contextMenuSettings).Render()[Model] List menuItems = new List(); menuItems.Add(new ContextMenuItem() { Id = "id" }); DiagramContextMenuSettings contextMenuSettings = new DiagramContextMenuSettings() { Show = true }; ViewBag.contextMenuSettings = contextMenuSettings;

DataSourceSettings

behavior	API in Essential JS 1	API in Essential JS 2
Defines the data source either as a collection of objects or as an instance of ej.Data Manager	Property: DataSourceSettings.DataSource [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.DataSource = getDataSource(); ViewData["diagramModel"] = Model;	Property: DataSourceSettings.DataManager [View] @Html.EJS().Diagram("container").DataSourceSettings(ViewBag.dataSource).Render()[Model] DiagramDataSource dataSource = new DiagramDataSource() { DataManager = items }; ViewBag.dataSource = dataSource;
Sets the unique id of the data	Property: DataSourceSettings.Id [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model]	Property: DataSourceSettings.Id [View] @Html.EJS().Diagram("container").DataSourceSettings(ViewBag.dataSource).Render()[Model]

source items	DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.Id = "ID"; ViewData["diagramModel"] = Model;	DiagramDataSource dataSource = new DiagramDataSource() { Id = "Id" }; ViewBag.dataSource = dataSource;
Defines the parent id of the data source item	Property: DataSourceSettings.Parent [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.Parent = "ReportingPerson"; ViewData["diagramModel"] = Model;	Property: DataSourceSettings.ParentId [View] @Html.EJS().Diagram("container").DataSourceSettings(ViewBag.dataSource).Render()[Model] DiagramDataSource dataSource = new DiagramDataSource() { ParentId = "ReportingPerson" }; ViewBag.dataSource = dataSource;
Describe s query to retrieve a set of data from the specified datasource	Property: DataSourceSettings.Query [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.Query = "datasource query"; ViewData["diagramModel"] = Model;	Not applicable
Sets the unique id of the root data source item	Property: DataSourceSettings.Root [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.Root = "E1"; ViewData["diagramModel"] = Model;	Property: DataSourceSettings.Root [View] @Html.EJS().Diagram("container").DataSourceSettings(ViewBag.dataSource).Render()[Model] DiagramDataSource dataSource = new DiagramDataSource() { Root = "E1" }; ViewBag.dataSource = dataSource;
Describe s the name of the table on which the specified query has to be	Property: DataSourceSettings.TableName [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.TableName =	Not applicable

execute d	"datasource table name"; ViewData["diagramModel"] = Model;	
Specifies the method name which is used to get the updated data from client side to the server side	Property: DataSourceSettings.CrudAction [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.CrudAction = { Read = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/GetNodes" } ViewData["diagramModel"] = Model;	Not applicable
Specifies the create method which is used to get the nodes to be added from client side to the server side	Property: DataSourceSettings.CrudAction.Create @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.CrudAction = { Create = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/AddNodes" } ViewData["diagramModel"] = Model;	Not applicable
Specifies the update method which is used to get the updated data from client side to the	Property: DataSourceSettings.CrudAction.Update [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.CrudAction = { Update = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/UpdateNodes" } ViewData["diagramModel"] = Model;	Not applicable

server side	es/api/Diagram/UpdateNodes" } ViewData["diagramModel"] = Model;	
Specifies the destroy method which is used to get the deleted items data from client side to the server side	Property: DataSourceSettings.CrudAction.Destroy [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.CrudAction = { Destroy = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/DeleteNodes" } ViewData["diagramModel"] = Model;	Not applicable
Specifies the read method to get the created nodes from client side to the server side	Property: DataSourceSettings.CrudAction.Read [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.CrudAction = { Read = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/GetNodes" } ViewData["diagramModel"] = Model;	Not applicable
Defines the data source either as a collection of objects or as an instance of ej.Data Manager	Property: DataSourceSettings.CustomFields [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.CustomFields = new List() { "Description", "Color" } ViewData["diagramModel"] = Model;	Property: DataSourceSettings.DataSource [View] @Html.EJS().Diagram("container").DataSourceSettings(ViewBag.dataSource).Render()[Model] DiagramDataSource dataSource = new DiagramDataSource() { Data = new List() { "Description", "Color" } }; ViewBag.dataSource = dataSource;

Defines the data source either as a collection of objects or as an instance of ej.Data Manager	Property: DataSourceSettings.ConnectionDataSource [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDataSource = new ConnectionDataSourceSettings() { DataSource = DiagramContext.HierarchicalDetails.ToList(), SourceNode = "SourceNode", TargetNode = "TargetNode", } ViewData["diagramModel"] = Model;	Not applicable
Sets the datasource for the connection datasource settings items	Property: DataSourceSettings.ConnectionDataSource.DataSource [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDataSource = new ConnectionDataSourceSettings() { DataSource = DiagramContext.HierarchicalDetails.ToList() } ViewData["diagramModel"] = Model;	Not applicable
Sets the unique id of the connection data source item	Property: DataSourceSettings.ConnectionDataSource.Id [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDataSource = new ConnectionDataSourceSettings() { Id = "id" } ViewData["diagramModel"] = Model;	Not applicable
Sets the source node of	Property: DataSourceSettings.ConnectionDataSource.SourceNode	Not applicable

the connecti on data source item	<pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDat aSource = new ConnectionDataSourceSettings() { SourceNode = "SourceNode" } ViewData["diagramModel"] = Model;</pre>	
Sets the target node of the connecti on data source item	<p>Property:DataSourceSettings.ConnectionDataSource.TargetNode</p> <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDat aSource = new ConnectionDataSourceSettings() { TargetNode = "TargetNode" } ViewData["diagramModel"] = Model;</pre>	Not applicable
Sets the sourceP ointX value of the connecti on data source item	<p>Property:DataSourceSettings.ConnectionDataSource.SourcePointX</p> <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDat aSource = new ConnectionDataSourceSettings() { SourcePointX = "200" } ViewData["diagramModel"] = Model;</pre>	Not applicable
Sets the sourceP ointY value of the connecti on data source item	<p>Property:DataSourceSettings.ConnectionDataSource.SourcePointY</p> <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDat</pre>	Not applicable

	<pre>aSource = new ConnectionDataSourceSettings() { SourcePointY = "300" } ViewData["diagramModel"] = Model;</pre>	
<p>Sets the targetPointX value of the connection data source item</p>	<p>Property: <code>DataSourceSettings.ConnectionDataSource.TargetPointX</code></p> <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionData aSource = new ConnectionDataSourceSettings() { TargetPointX = "100" } ViewData["diagramModel"] = Model;</pre>	Not applicable
<p>Sets the targetPointY value of the connection data source item</p>	<p>Property: <code>DataSourceSettings.ConnectionDataSource.TargetPointY</code></p> <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionData aSource = new ConnectionDataSourceSettings() { TargetPointY = "100" } ViewData["diagramModel"] = Model;</pre>	Not applicable
<p>Specifies the method name which is used to get updated connectors from client side to the server side</p>	<p>Property: <code>DataSourceSettings.ConnectionDataSource.CrudAction</code></p> <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionData aSource = new ConnectionDataSourceSettings() { CrudAction = { Create = "http://js.Syncfusion.Com/demos/ejservic es/api/Diagram/AddConnectors" } } ViewData["diagramModel"] = Model;</pre>	Not applicable

Specifies the create method which is used to get the connectors to be added from client side to the server side	Property: DataSourceSettings.ConnectionDataSource.CrudAction.Create <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDataSource = new ConnectionDataSourceSettings() { CrudAction = { Create = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/AddConnectors" } } ViewData["diagramModel"] = Model;</pre>	Not applicable
Specifies the update method which is used to get the updated connectors from client side to the server side	Property: DataSourceSettings.ConnectionDataSource.CrudAction.Update <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDataSource = new ConnectionDataSourceSettings() { CrudAction = { Update = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/UpdateConnectors" } } ViewData["diagramModel"] = Model;</pre>	Not applicable
Specifies the destroy method which is used to get the deleted items data from client side to the	Property: DataSourceSettings.ConnectionDataSource.CrudAction.Destroy <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionDataSource = new ConnectionDataSourceSettings() { CrudAction = { Destroy = "http://js.Syncfusion.Com/demos/ejservices/api/Diagram/DestroyConnectors" } } ViewData["diagramModel"] = Model;</pre>	Not applicable

server side	es/api/Diagram/DeleteConnectors" } } ViewData["diagramModel"] = Model;	
Specifies the read method which is used to get the data from client side to the server side	Property: DataSourceSettings.ConnectionData DataSource.CrudAction.Read [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionData aSource = new ConnectionDataSourceSettings() { CrudAction = { Read = "http://js.Syncfusion.Com/demos/ejservic es/api/Diagram/GetConnectors" } } ViewData["diagramModel"] = Model;	Not applicable
Specifies the custom fields to get the updated data from client side to the server side	Property: DataSourceSettings.ConnectionData DataSource.CustomFields [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties).Render();[Model] DiagramProperties Model = new DiagramProperties(); Model.DataSourceSettings.ConnectionData aSource = new ConnectionDataSourceSettings() { CustomFields = new List() { "Description", "Color", } } ViewData["diagramModel"] = Model;	Not applicable

DefaultSettings

behavior	API in Essential JS 1	API in Essential JS 2
Initializes the default values for nodes and	Property: DefaultSettings.Node @{ Node defaultNode = new Node() { FillColor = "red" }; Html.EJ().Diagram("diagram").DefaultSettings(d =>d.Node(defaultNode)).Render(); }	Property: GetNodeDefaults @Html.EJS().Diagram("container").GetNodeDefa ults("getNodeDefaults") function getNodeDefaults(obj, diagram) { obj.Shape = { type: 'Basic', shape: 'Rectangle', cornerRadius: 10 }; obj.Width = 80; obj.Style.StrokeWidth = 2; obj.Style.StrokeColor = '#6F409F'; obj.Height = 35; }

connectors		
Initializes the default connector properties	Property: DefaultSettings.Connector <pre>@{ Connector defaultConnector = new Connector() { LineColor = "red", LineWidth = 4, LineDashArray = "2,2" }; Html.EJ().Diagram("diagram").DefaultSettings(d =>d.Connector(defaultConnector)).Render(); }</pre>	Property: GetConnectorDefaults <pre>@Html.EJS().Diagram("container").GetConnectorDefaults("getConnectorDefaults").Render() function getConnectorDefaults(obj, diagram) { obj.Type = 'Bezier'; obj.Style.StrokeColor = '#6f409f'; obj.Style.StrokeWidth = 2; obj.TargetDecorator = { style: { strokeColor: '#6f409f', fill: '#6f409f', } } }</pre>
Initializes the default properties of groups	Property: DefaultSettings.Group <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Group defaultNode = new Group() { Constraints = NodeConstraints.Default ~NodeConstraints.Drag }; Model.DefaultSettings.Group = defaultNode; ViewData["diagramModel"] = Model;</pre>	Not applicable

DrawType

behavior	API in Essential JS 1	API in Essential JS 2
Sets the type of JSON object to be drawn through drawing tool	Property: DrawType <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Model.DrawType = { Type: "node" }; ViewData["diagramModel"] = Model;</pre>	Property: DrawingObject <pre>@Html.EJS().Diagram("container").Created("diagramCreated").Render() function diagramCreated() { diagram = document.getElementById("diagram").Ej2_instances[0]; diagram.DrawingObject = { shape: { type: 'Basic', shape: 'Rectangle' } }; diagram.Tool = ej.Diagrams.DiagramTools.ContinuousDraw; diagram.DataBind(); }</pre>

EnableAutoScroll

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables auto scroll in diagram	Property: EnableAutoScroll <pre>@{ Html.EJ().Diagram("diagram").EnableAutoScroll(false).Render(); }</pre>	Property: CanAutoScroll <pre>@Html.EJS().Diagram("container").ScrollSettings(new DiagramScrollSettings() { CanAutoScroll = true}).Render()</pre>

EnableContextMenu

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables diagram context menu	Property: EnableContextMenu <pre>@{ Html.EJ().Diagram("diagram").EnableContextMenu(true).Render(); }</pre>	Property: ContextMenuSettings.Show <pre>@Html.EJS().Diagram("container").ContextMenuSettings(new DiagramContextMenuSettings() { Show = true}).Render()</pre>

GetCustomCursor

behavior	API in Essential JS 1	API in Essential JS 2
Enable or disable rendering component with custom cursor	Not applicable	Property: GetCustomCursor <pre>@Html.EJS().Diagram("container").GetCustomCursor("getCustomCursor").Render() function getCustomCursor(action, active) { var cursor; if (active && action === 'Drag') { cursor = '-webkit-grabbing'; } else if (action === 'Drag') { cursor = '-webkit-grabbing'; } return cursor; }</pre>

GetCustomProperty

behavior	API in Essential JS 1	API in Essential JS 2

Allows to get the custom properties that have to be serialized	Not applicable	Property: GetCustomProperty @Html.EJS().Diagram("container").GetCustomProperty("getCustomProperty").Render() function getCustomProperty(key: string) { if (key === 'nodes') { return ['description']; } return null; }
--	----------------	--

GetDescription

behavior	API in Essential JS 1	API in Essential JS 2
Allows to get the custom description	Not applicable	Property: GetDescription @Html.EJS().Diagram("container").GetDescription("getAccessibility").Render() function getAccessibility(object, diagram) { var value; if (object.propName == "connectors") { value = 'clicked on Connector'; } else { value = undefined; } return value; }

GetCustomTool

behavior	API in Essential JS 1	API in Essential JS 2
Allows to get the custom tool	Not applicable	Property: GetCustomTool @Html.EJS().Diagram("container").Width("1000").Height("645px").GetCustomTool("getTool") function getTool(action) { var tool; if (action === 'userHandle') { tool = new CloneTool(diagram.CommandHandler, true); } return tool; } class CloneTool extends ToolBase { public mouseDown(args) { super.MouseDown(args); diagram.Copy(); diagram.Paste(); } }

Height

behavior	API in Essential JS 1	API in Essential JS 2
Specifies the height of the diagram	Property: Height @{ Html.EJ().Diagram("diagram").Height("500px").Render(); }	Property: Height @Html.EJS().Diagram("container").Height("645px").Render()

HistoryManager

behavior	API in Essential JS 1	API in Essential JS 2
A method that takes a history entry as argument and returns whether the specific entry can be popped or not	Property:HistoryManager.CanPop	Not applicable
A method that ends grouping the changes	Property:HistoryManager.CloseGroupAction	
A method that removes the history of a recent change made in diagram	Property:HistoryManager.Pop	Not applicable
A method that allows to track the custom	Property:HistoryManager.Push	Property:HistoryList.Push

changes made in diagram		
Defines what should be happened while trying to restore a custom change	Property: HistoryManager.Redo <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Model.HistoryManager = new HistoryManager() { Redo = "customUndoRedo" } ViewData["diagramModel"] = Model; function customUndoRedo(args) { var diagram = \$("#diagramcontent").EjDiagram("instance"); var node = args.Object; var currentState = node.employeeData; node.employeeData = args.PrevState; args.PrevState = currentState; }</pre>	Property: HistoryList.Redo
Gets the number of redo actions to be stored on the history manager. Its an read-only property and the collection should not be modified	Property: HistoryManager.RedoStack	Property: HistoryList.RedoStack
Restricts the undo and redo actions to a certain limit	Property: HistoryManager.StackLimit	Not applicable
A method	Property: HistoryManager.StartGroupAction	Property: HistoryList.StartGroupAction

that starts to group the changes to revert/restore them in a single undo or redo		
Defines what should be happened while trying to revert a custom change	Property: <code>HistoryManager.Undo</code> <code>@{ Html.EJ().Diagram("diagram").HistoryManager(h=>h.Undo("customUndoRedo").Redo("customUndoRedo")).Render(); }</code>	Property: <code>HistoryList.Undo</code>
Gets the number of undo actions to be stored on the history manager. Its an read-only property and the collection should not be modified	Property: <code>HistoryManager.UndoStack</code>	Property: <code>HistoryList.UndoStack</code>
Set the current	Not applicable	Property: <code>HistoryList.CurrentEntry</code>

entry object		
Set the history entry can be undo	Not applicable	Property: <code>HistoryList.CanUndo</code>
Set the history entry can be redo	Not applicable	Property: <code>HistoryList.CanRedo</code>
Used to decide to stored the changes to history	Property: <code>HistoryManager.CanLog</code> <pre>var diagram = \$("#diagramcontent").EjDiagram("instance"); diagram.Model.HistoryManager.CanLog();</pre>	Property: <code>HistoryList.CanLog</code>

Layout

behavior	API in Essential JS 1	API in Essential JS 2
Specifies the custom bounds to arrange/align the layout	Property: <code>Layout.Bounds</code> <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.Bounds(new Rectangle() { X = 10, Y = 10, Width = 100, Height = 100 })).Render(); }</pre>	Property: <code>Layout.Bounds</code> <pre>@Html.EJS().Diagram("container") .Layout(l => l.Bounds("getBounds")).Render();</pre>
Defines the fixed node with reference to which, the layout will be arranged and fixed node will not be repositioned	Property: <code>Layout.FixedNode</code> <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.FixedNode("no deName")).Render(); }</pre>	Property: <code>Layout.FixedNode</code> <pre>@Html.EJS().Diagram("container") .Layout(l => l.FixedNode("node")).Render();</pre>

Customizes the orientation of trees/sub trees	Property: Layout.GetLayoutInfo <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.GetLayoutInfo("getLayoutInfo")).Render(); }</pre>	Property: Layout.GetLayoutInfo <pre>@Html.EJS().Diagram("container") .Layout(l => l.GetLayoutInfo("getLayoutInfo")). Render();</pre>
Defines a method to customize the segments based on source and target nodes	Property: Layout.GetConnectorSegments <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.GetConnectorSegments("getConnectorSegments")).Render(); }</pre>	Property: Layout.ConnectorSegments <pre>@Html.EJS().Diagram("container") .Layout(l => l.ConnectorSegments(ConnectorSegments.Default)).Render();</pre>
Sets the space to be horizontally left between nodes	Property: Layout.HorizontalSpacing <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.HorizontalSpacing(50)).Render(); }</pre>	Property: Layout.HorizontalSpacing <pre>@Html.EJS().Diagram("container") .Layout(l => l.HorizontalSpacing(50)).Render();</pre>
Defines the space to be left between layout bounds and layout	Property: Layout.Margin <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.Margin(new Margin() { Left = 20, Right = 20, Top = 20, Bottom = 20 })).Render(); }</pre>	Property: Layout.Margin <pre>@Html.EJS().Diagram("container") .Layout(l => l.Margin(m=> m.Top(50).Bottom(0).Left(50).Right(0))).Render();</pre>
Defines how to horizontally align the layout within the layout bounds	Property: Layout.HorizontalAlignment <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.HorizontalAlignment(HorizontalAlignment.Center)).Render(); }</pre>	Property: Layout.HorizontalAlignment <pre>@Html.EJS().Diagram("container") .Layout(l => l.HorizontalAlignment(HorizontalAlignment.Center)).Render();</pre>
Defines how to vertically align the layout within the layout bounds	Property: Layout.VerticalAlignment <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.VerticalAlignment(VerticalAlignment.Center)).Render(); }</pre>	Property: Layout.VerticalAlignment <pre>@Html.EJS().Diagram("container") .Layout(l => l.VerticalAlignment(VerticalAlignment.Top)).Render();</pre>

Sets the orientation /direction to arrange the diagram elements	Property: Layout.Orientation <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.Orientation(LayoutOrientation.LeftToRight)).Render(); }</pre>	Property: Layout.Orientation <pre>@Html.EJS().Diagram("container") .Layout(l => l.Orientation(LayoutOrientation.TopToBottom)).Render();</pre>
Sets the type of the layout based on which the elements will be arranged	Property: Layout.Type <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.Type(LayoutTypes.HierarchicalTree)).Render(); }</pre>	Property: Layout.Type <pre>@Html.EJS().Diagram("container") .Layout(l => l.Type(LayoutType.OrganizationalChart)).Render();</pre>
Sets the space to be vertically left between nodes	Property: Layout.VerticalSpacing <pre>@{ Html.EJ().Diagram("diagram").Layout(l=>l.VerticalSpacing(50)).Render(); }</pre>	Property: Layout.VerticalSpacing <pre>@Html.EJS().Diagram("container") .Layout(l => l.VerticalSpacing(50)).Render();</pre>
Sets the value is used to define the root node of the layout	Property: Layout.Root <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Model.Layout.Root = "nodeName"; ViewData["diagramModel"] = Model;</pre>	Property: Layout.Root <pre>@Html.EJS().Diagram("container") .Layout(l => l.Root("nodeName")).Render();</pre>
Defines how long edges should be, ideally. This will be the resting length for the springs	Property: Layout.SpringFactor <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Model.Layout.SpringFactor = .4F; ViewData["diagramModel"] = Model;</pre>	Property: Layout.SpringFactor <pre>@Html.EJS().Diagram("container") .Layout(l => l.SpringFactor(0.8)).Render();</pre>
Defines how long edges should be, ideally. This will be the resting	Property: Layout.MaxIteration <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties();</pre>	Property: Layout.MaxIteration <pre>@Html.EJS().Diagram("container") .Layout(l => l.MaxIteration(500)).Render();</pre>

length for the springs	Model.Layout.MaxIteration = 50; ViewData["diagramModel"] = Model;	
Defines how long edges should be, ideally. This will be the resting length for the springs	Property: Layout.SpringLength [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties)[Model] DiagramProperties Model = new DiagramProperties(); Model.Layout.SpringLength = 50; ViewData["diagramModel"] = Model;	Property: Layout.SpringLength @Html.EJS().Diagram("container") .Layout(l => l.SpringLength(80)).Render();
Sets how to define the connection direction (first segment direction & last segment direction)	Not applicable	Property: Layout.ConnectionDirection @Html.EJS().Diagram("container") .Layout(l => l.ConnectionDirection(Connection Direction.Auto)).Render();
Enables/Disables animation option when a node is expanded/collapsed	Not applicable	Property: Layout.EnableAnimation @Html.EJS().Diagram("container") .Layout(l => l.EnableAnimation(true)).Render() ;
Defines whether an object should be at the left/right of the mind map. Applicable only for the direct children of the root node	Not applicable	Property: Layout.GetBranch @Html.EJS().Diagram("container") .Layout(l => l.GetBranch("getBranch")).Render (); function getBranch(node, nodes) { return node.Data.Branch; }

Nodes

behavior	API in Essential JS 1	API in Essential JS 2
Array of JSON objects where each object represents a node	Property: Nodes @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Name = "node1" })).Render(); }	Property: Nodes [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Id = "nodeName" }); ViewBag.Nodes = nodes;
Defines the type of BPMN Activity. Applicable, if the node is a BPMN activity	Property: Nodes.Activity @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Activity = BPMNActivity.SubProcess })).Render(); }	Property: Nodes.Shape.Activity [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Id = "nodeName", shape: { type = 'Bpmn', shape = 'Activity', activity = { activity = "Task" } } }); ViewBag.Nodes = nodes;
To maintain additional information about nodes	Property: Nodes.AddInfo @{ Dictionary AddInfo = new Dictionary(); AddInfo.Add("Description", "Bidirectional Flow"); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { AddInfo = AddInfo })).Render(); }	Property: Nodes.AddInfo [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] Dictionary AddInfo = new Dictionary(); AddInfo.Add("Description", "Bidirectional Flow"); List nodes = new List(); nodes.Add(new DiagramNode() { AddInfo = AddInfo }); ViewBag.Nodes = nodes;
Defines the additional information of a process. It is not directly related to the message flows or sequence flows of the process	Property: Nodes.Annotation @{ Html.EJ().Diagram("diagram")Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation() { Text= "Text" } })).Render(); }	Property: Nodes.Shape.Annotations [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramNodeAnnotation() { Content = "Place Order" } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;
Sets the angle	Property: Nodes.Annotation.Angle	Property: Nodes.Shape.Annotations.Angle

between the BPMN shape and the annotation	<pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation(){ Angle=-45 } })).Render(); }</pre>	<pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramBpmnAnnotation() { Angle = -45 }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;</pre>
Sets the direction of the text annotation	<p>Property: Nodes.Annotation.Direction</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation(){Direction=BPMNAnnotationDirections.Left } })).Render(); }</pre>	Not applicable
Sets the height of the text annotation	<p>Property: Nodes.Annotation.Height</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation(){ Height=50 } })).Render(); }</pre>	<pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramBpmnAnnotation() { Height = 50 }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;</pre>
Sets the distance between the BPMN shape and the annotation	<p>Property: Nodes.Annotation.Length</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation(){ Length=150 } })).Render(); }</pre>	<p>Property: Nodes.Shape.Annotations.Length</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramBpmnAnnotation() { Length = 150 }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;</pre>
Defines the additional information about the flow object in a BPMN Process	<p>Property: Nodes.Annotation.Text</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation(){ Text= "Text" } })).Render(); }</pre>	<p>Property: Nodes.Shape.Annotations.Text</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramBpmnAnnotation() { Text= "Text" }); List nodes = new List(); nodes.Add(new</pre>

		DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;
Sets the width of the text annotation	Property: Nodes.Annotation.Width @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Annotation = new BPMNAnnotation(){ Width=100 } })).Render(); }	Property: Nodes.Shape.Annotations.Width [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramBpmnAnnotation() { Width = 100 }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;
Sets the id for the annotation	Not applicable	Property: Nodes.Shape.Annotations.Id [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List Node = new List(); Node.Add(new DiagramBpmnAnnotation() { Id = "id" }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = Node}); ViewBag.Nodes = nodes;
Defines whether the group can be ungrouped or not	Property: Nodes.CanUngroup [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Node node1 = new Node() { Name = "node1"}; Node node2 = new Node() { Name = "node2"}; Group group = new Group(); group.Children.Add(node1); group.Children.Add(node2); group.CanUngroup = false; ViewData["diagramModel"] = Model;	Not applicable
Array of JSON objects where each object represents a child node/connector	Property: Nodes.Children [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Node node1 = new Node() { Name = "node1"}; Node node2 = new Node() { Name = "node2"}; Group group = new	

	<pre>Group(); group.Children.Add(node1); group.Children.Add(node2); ViewData["diagramModel"] = Model;</pre>	
Sets the type of UML classifier	Property: Nodes.Classifier <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new UMLClassifier() { Classifier=ClassifierShapes.Class })).Render(); }</pre>	Not applicable
Defines the name, attributes and methods of a Class. Applicable, if the node is a Class	Property: Nodes.Class <pre>@{ Html.EJ().Diagram("diagram").Nodes(n=>n.Add (new UMLClassifier() { Classifier = ClassifierShapes.Class, Class = { Name = "name" } })).Render(); }</pre>	Not applicable
Sets the name of class	Property: Nodes.Class.Name <pre>@{ Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Name = "name" } })).Render(); }</pre>	Not applicable
Defines the collection of attributes	Property: Nodes.Class.Attributes <pre>@{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Name = "name" } }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Attributes= attribute } })).Render(); }</pre>	Not applicable
Sets the name of the attribute	Property: Nodes.Class.Attributes.Name <pre>@{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Name = "name" } }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Attributes= attribute } })).Render(); }</pre>	Not applicable
Sets the data type of attribute	Property: Nodes.Class.Attributes.Type <pre>@{ Collection attribute = new Collection();</pre>	Not applicable

	<pre>attribute.Add(new UMLAttribute() { Type = "Date" }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Attributes= attribute} })).Render(); }</pre>	
Defines the visibility of the attribute	<pre>Property:Nodes.Class.Attributes.Scope @{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Scope = Scopes.Protected }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Attributes= attribute} })).Render(); }</pre>	Not applicable
Defines the collection of methods of a Class	<pre>Property:Nodes.Class.Methods @{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Name = { "getHistory" } }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }</pre>	Not applicable
Sets the name of the method	<pre>Property:Nodes.Class.Methods.Name @{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Name = "getHistory" }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }</pre>	Not applicable
Defines the arguments of the method	<pre>Property:Nodes.Class.Methods.Arguments @{ Collection parameter = new Collection(); parameter.Add(new UMLParameter() { Name = "Date", Type = "String" }); Collection methods = new Collection(); methods.Add(new UMLMethod() { Parameters = parameter }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() {Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }</pre>	Not applicable

Defines the name, attributes and methods of a Class. Applicable, if the node is a Class	Property: Nodes.Class.Methods.Arguments.Name @{ Collection parameter = new Collection(); parameter.Add(new UMLParameter() { Name = "Date" }); Collection methods = new Collection(); methods.Add(new UMLMethod() { Parameters = parameter }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() { Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }	Not applicable
Sets the type of the argument	Property: Nodes.Class.Methods.Arguments.Type @{ Collection parameter = new Collection(); parameter.Add(new UMLParameter() { Type = "String" }); Collection methods = new Collection(); methods.Add(new UMLMethod() { Parameters = parameter }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() { Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }	Not applicable
Sets the return type of the method	Property: Nodes.Class.Methods.Type @{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Type = "History" }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() { Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }	Not applicable
Sets the visibility of the method	Property: Nodes.Class.Methods.Scope @{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Scope = Protected }); Html.EJ().Diagram("diagram.Nodes(n => n.Add(new UMLClassifier() { Classifier=ClassifierShapes.Class, Class= { Methods = methods} })).Render(); }	Not applicable
Defines the state of the node is	Property: Nodes.CollapseIcon	Property: Nodes.CollapseIcon

collapsed/expanded	<pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { Shape=IconShapes.ArrowUp, } })}).Render(); }</pre>	<pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { Shape = IconShapes.ArrowUp } }); ViewBag.Nodes = nodes;</pre>
Sets the border color for collapse icon of node	<p>Property: Nodes.CollapseIcon.BorderColor</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { BorderColor= "red" } })}).Render(); }</pre>	<p>Property: Nodes.CollapseIcon.BorderColor</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { BorderColor = "red" } }); ViewBag.Nodes = nodes;</pre>
Sets the border width for collapse icon of node	<p>Property: Nodes.CollapseIcon.BorderWidth</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { BorderWidth=2 } })}).Render(); }</pre>	<p>Property: Nodes.CollapseIcon.BorderWidth</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { BorderWidth = 2 } }); ViewBag.Nodes = nodes;</pre>
Sets the fill color for collapse icon of node	<p>Property: Nodes.CollapseIcon.FillColor</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { FillColor="green" } })}).Render(); }</pre>	<p>Property: Nodes.CollapseIcon.Fill</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { Fill = "red" } }); ViewBag.Nodes = nodes;</pre>
Defines the height for collapse icon of node	<p>Property: Nodes.CollapseIcon.Height</p> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { Height = 10 } })}).Render(); }</pre>	<p>Property: Nodes.CollapseIcon.Height</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { Height = 10 } }); ViewBag.Nodes = nodes;</pre>

Sets the horizontal alignment of the icon	Property: Nodes.CollapseIcon.HorizontalAlignment <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { HorizontalAlignment = HorizontalAlignment.Left } })).Render(); }</pre>	Property: Nodes.CollapseIcon.HorizontalAlignment <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { HorizontalAlignment = HorizontalAlignment.Left } }); ViewBag.Nodes = nodes;</pre>
To set the margin for the collapse icon of node	Property: Nodes.CollapseIcon.Margin <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { Margin = new Margin() { Left = 5 } } })).Render(); }</pre>	Property: Nodes.CollapseIcon.Margin <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { Margin = new DiagramMargin() { Left = 5 } } }); ViewBag.Nodes = nodes;</pre>
Sets the fraction/ratio (relative to node) that defines the position of the icon	Property: Nodes.CollapseIcon.Offset <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = {Offset=new DiagramPoint() { X= 0, Y= 0.5F } } })).Render(); }</pre>	Property: Nodes.CollapseIcon.Offset <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { Offset = { X = 0, Y = 0.5 } } }); ViewBag.Nodes = nodes;</pre>
Defines the shape of the collapsed state of the node	Property: Nodes.CollapseIcon.Shape <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = { Shape = IconShapes.ArrowUp } })).Render(); }</pre>	Property: Nodes.CollapseIcon.Shape <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { CollapseIcon = { Shape = IconShapes.ArrowUp } }); ViewBag.Nodes = nodes;</pre>
Sets the vertical alignment of the icon	Property: Nodes.CollapseIcon.VerticalAlignment <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CollapseIcon = {</pre>	Property: Nodes.CollapseIcon.VerticalAlignment <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List</pre>

	<code>VerticalAlignment= VerticalAlignment.Top } }).Render(); }</code>	<code>nodes = new List(); nodes.Add(new DiagramNode() { Collapselcon = { VerticalAlignment= VerticalAlignment.Top } }); ViewBag.Nodes = nodes;</code>
Defines the custom content of the icon	Not applicable	Property: <code>Nodes.Collapselcon.Content</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Collapselcon = { Shape = IconShapes.Template, Content = " + " } }); ViewBag.Nodes = nodes;</code>
Defines the geometry of a path	Not applicable	Property: <code>Nodes.Collapselcon.PathData</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Collapselcon = { Shape = IconShapes.Path, PathData = "M0,0 L0,100" } }); ViewBag.Nodes = nodes;</code>
Defines the corner radius of the icon border	Not applicable	Property: <code>Nodes.Collapselcon.CornerRadius</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Collapselcon = { CornerRadius = 5 } }); ViewBag.Nodes = nodes;</code>
Defines the space that the icon has to be moved from the icon border	Not applicable	Property: <code>Nodes.Collapselcon.Padding</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Collapselcon = { Padding = { Left = 15 } } }); ViewBag.Nodes = nodes;</code>
Defines the distance to be left between a	Property: <code>Nodes.ConnectorPadding</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n =></code>	Not applicable

node and its connections (incoming and outgoing connections)	<code>n.Add(new Node() { ConnectorPadding = 5 }).Render();</code>	
Enables or disables the default behaviors of the node	Property: <code>Nodes.Constraints</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Constraints = NodeConstraints.Default })).Render(); }</code>	Property: <code>Nodes.Constraints</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Constraints = NodeConstraints.Default }); ViewBag.Nodes = nodes;</code>
Defines the orientation of the container. Applicable, if the group is a container	Property: <code>Nodes.Container.Orientation</code> [View] <code>@Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Node node1 = new Node() { Name = "node1"}; Node node2 = new Node() { Name = "node2"}; Group group = new Group(); group.Children.Add(node1); group.Children.Add(node2); group.Container = new Container() { Orientation = "Horizontal" }; ViewData["diagramModel"] = Model;</code>	Not applicable
Sets the type of the container. Applicable if the group is a container.	Property: <code>Nodes.Container.Type</code> [View] <code>@Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Node node1 = new Node() { Name = "node1"}; Node node2 = new Node() { Name = "node2"}; Group group = new Group(); group.Children.Add(node1); group.Children.Add(node2); group.Container = new Container() { Orientation = "Horizontal" }; ViewData["diagramModel"] = Model;</code>	Not applicable
Defines the corner radius of	Property: <code>Nodes.CnerRadius</code> <code>@{ Html.EJ().Diagram("diagram")Nodes(n =></code>	Not applicable

rectangular shapes	<code>n.Add(new BasicShape() { CornerRadius=5 })).Render(); }</code>	
This property allows you to customize nodes appearance using user-defined CSS	Property: <code>Nodes.CssClass</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { CssClass= "hoverNode" })).Render(); }</code>	Not applicable
Defines the BPMN data object	Property: <code>Nodes.Data.Type</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.DataObject, Data = new BPMNDataObject() { Type = BPMNDataObjects.Input } })).Render(); }</code>	Property: <code>Nodes.Shape.DataObject.Type</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "BPMN", Shape = "DataObject", dataObject = new DiagramBpmnDataObject() { Type = BpmnDataObjects.Input } } }); ViewBag.Nodes = nodes;</code>
Defines whether the BPMN data object is a collection or not	Property: <code>Nodes.Data.Collection</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.DataObject, Data = new BPMNDataObject() { Collection = false } })).Render(); }</code>	Property: <code>Nodes.Shape.DataObject.Collection</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "BPMN", Shape = "DataObject", dataObject = new DiagramBpmnDataObject() { Collection = false } } }); ViewBag.Nodes = nodes;</code>
Defines an Enumeration in a UML Class Diagram	Property: <code>Nodes.Enumeration</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new UMLClassifier() { Classifier = ClassifierShapes.Enumeration, Enumeration = { Name = "AccountType" } })).Render(); }</code>	Not applicable
Sets the name of the Enumeration	Property: <code>Nodes.Enumeration.Name</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new UMLClassifier() { Classifier =</code>	Not applicable

	ClassifierShapes.Enumeration, Enumeration={ Name = "AccountType" } })).Render(); }	
Defines the collection of enumeration members	Property: Nodes.Enumeration.Members @{ Collection member = new Collection(); member.Add(new UMLEnumeration() { Name = "CheckingAccount" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new UMLClassifier() { Classifier = ClassifierShapes.Enumeration, Enumeration = new UMLEnumeration() { Members = member } })).Render(); }	Not applicable
Sets the name of the enumeration member	Property: Nodes.Enumeration.Members.Name @{ Collection member = new Collection(); member.Add(new UMLEnumeration() { Name = "CheckingAccount" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new UMLClassifier() { Classifier = ClassifierShapes.Enumeration, Enumeration = new UMLEnumeration() { Members = member } })).Render(); }	Not applicable
Sets the type of the BPMN Events. Applicable, if the node is a BPMN event	Property: Nodes.Event @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape=BPMNShapes.Event, Event=BPMNEvents.Intermediate })).Render(); }	Property: Nodes.Shape.Event [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Event", Event = { Event = BpmnEvents.Start } } }); ViewBag.Nodes = nodes;
Defines the type of the trigger	Property: Nodes.Event.Trigger @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.Event, Trigger = BPMNTriggers.None })).Render(); }	Property: Nodes.Shape.Event.Trigger [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Event", Event = { Trigger =

		BpmnTriggers.None } } }); ViewBag.Nodes = nodes;
Defines whether the node can be automatically arranged using layout or not	Property: Nodes.ExcludeFromLayout @{ Html.EJ().Diagram("diagram").Layout(l=>l.Type(LayoutTypes.HierarchicalTree)).Nodes(n => n.Add(new Node() { Name = "Patient"ExcludeFromLayout=true })).Add(new Node() { Name = "Patient1" }).Add(new Node() { Name = "Patient2" })).Render(); }	Property: Nodes.ExcludeFromLayout [View] @Html.EJS().Diagram("container").Layout(new DiagramLayout() { Type = LayoutType.HierarchicalTree}).Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Id = "Patient", ExcludeFromLayout= true } } }); nodes.Add(new DiagramNode() { Id = "Patient1" } } }); nodes.Add(new DiagramNode() { Id = "Patient2" } } }); ViewBag.Nodes = nodes;
Defines the fill color of the node	Property: Nodes.FillColor @{ Html.EJ().Diagram("diagram").Height("500px").Width("500px").Nodes(n => n.Add(new Node() { FillColor="red" })).Render(); }	Property: Nodes.Style.Fill [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Fill = "red" } } }); ViewBag.Nodes = nodes;
Sets the type of the BPMN Gateway. Applicable, if the node is a BPMN gateway	Property: Nodes.Gateway @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape=BPMNShapes.Gateway, Gateway=BPMNGateways.Exclusive })).Render(); }	Property: Nodes.Shape.Gateway [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Gateways", gateWay = { Type = BpmnGateways.Exclusive } } } }); ViewBag.Nodes = nodes;
Paints the node with linear color transitions	Property: Nodes.Gradient.Type @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new LinearGradient() { Type = "linear" } })).Render(); }	Property: Nodes.Style.Gradient.Type [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Gradient = new DiagramGradient() { Type =

		GradientType.Linear } } }); ViewBag.Nodes = nodes;
Defines the x1 value of linear gradient	Property:Nodes.Gradient.LinearGradient.X1 @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new LinearGradient() { Type = "linear", X1 = 0 } })).Render(); }	Property:Nodes.Style.Gradient.LinearGradient.X1
Defines the x2 value of linear gradient	Property:Nodes.Gradient.LinearGradient.X2 @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new LinearGradient() { Type = "linear", X2 = 50 } })).Render(); }	Property:Nodes.Style.Gradient.LinearGradient.X2
Defines the y1 value of linear gradient	Property:Nodes.Gradient.LinearGradient.Y1 @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new LinearGradient() { Type = "linear", Y1 = 0 } })).Render(); }	Property:Nodes.Style.Gradient.LinearGradient.Y1
Defines the y2 value of linear gradient	Property:Nodes.Gradient.LinearGradient.Y2 @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new LinearGradient() { Type = "linear", Y2 = 50 } })).Render(); }	Property:Nodes.Style.Gradient.LinearGradient.Y2
Defines the type of gradient	Property:Nodes.Gradient.RadialGradient.Type @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Type = "radial" } })).Render(); }	Property:Nodes.Style.Gradient.Type [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Gradient = new DiagramGradient() { Type = GradientType.Radial } } }); ViewBag.Nodes = nodes;
Defines the position of the outermost circle	Property:Nodes.Gradient.RadialGradient.Cx @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Type = "radial", CX=50 } })).Render(); }	Property:Nodes.Style.RadialGradient.Cx

Defines the outer most circle of the radial gradient	Property: Nodes.Gradient.RadialGradient.Cy @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Type = "radial", CY=50 } })).Render(); }	Property: Nodes.Style.RadialGradient.Cy
Defines the innermost circle of the radial gradient	Property: Nodes.Gradient.RadialGradient.Fx @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Type = "radial", FX=50 } })).Render(); }	Property: Nodes.Style.RadialGradient.Fx
Defines the innermost circle of the radial gradient	Property: Nodes.Gradient.RadialGradient.Fy @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Type = "radial", FY=50 } })).Render(); }	Property: Nodes.Style.RadialGradient.Fy
Defines the different colors and the region of color transitions	Property: Nodes.Gradient.RadialGradient.Stops @{ Collection stops = new Collection(); stops.Add(new Stop() { Color = "white", Offset = 0 }); stops.Add(new Stop() { Color = "red", Offset = 50 }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Stops = stops } })).Render(); }	Property: Nodes.Style.RadialGradient.Stops [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List stops = new List(); stops.Add(new DiagramStop() { Color = "white", Offset = 0 }); stops.Add(new DiagramStop() { Color = "red", Offset = 50 }); List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Gradient = new DiagramGradient() { Stops = stops } } }); ViewBag.Nodes = nodes;
Sets the color to be filled over the specified region	Property: Nodes.Gradient.Stops.Color @{ Collection stops = new Collection(); stops.Add(new Stop() { Color = "white" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Stops = stops } })).Render(); }	Property: Nodes.Style.Gradient.Stops.Color [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List stops = new List(); stops.Add(new DiagramStop() { Color = "white" }); List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Gradient = new

		DiagramGradient() { Stops = stops } } } ViewBag.Nodes = nodes;
Sets the position where the previous color transition ends and a new color transition starts	Property: Nodes.Gradient.Stops.Offset @{ Collection stops = new Collection(); stops.Add(new Stop() { Offset = 0 }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Stops = stops } })).Render(); }	Property: Nodes.Style.Gradient.Stops.Offset [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List stops = new List(); stops.Add(new DiagramStop() { Offset = 0 }); List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Gradient = new DiagramGradient() { Stops = stops } } }); ViewBag.Nodes = nodes;
Describes the transparency level of the region	Property: Nodes.Gradient.Stops.Opacity @{ Collection stops = new Collection(); stops.Add(new Stop() { Opacity = 0.5F }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Gradient= new RadialGradient() { Stops = stops } })).Render(); }	Property: Nodes.Style.Gradient.Stops.Opacity [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List stops = new List(); stops.Add(new DiagramStop() { Opacity = 0.5 }); List nodes = new List(); nodes.Add(new DiagramNode() { Style = new DiagramShapeStyle { Gradient = new DiagramGradient() { Stops = stops } } }); ViewBag.Nodes = nodes;
Defines the header of a swimlane/lane	Property: Nodes.Header [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); SwimLane swimLane = new SwimLane(); Header header = new Header(); header.Text = "Text"; swimLane.Header = header; ~ Model.Nodes.Add(swimLane); ViewData["diagramModel"] = Model;	Not applicable
Defines the height of the node	Property: Nodes.Height @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Height=100 })).Render(); }	Property: Nodes.Height [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List

		<pre>nodes = new List(); nodes.Add(new DiagramNode() { Height = 100 }); ViewBag.Nodes = nodes;</pre>
Sets the horizontal alignment of the node. Applicable, if the parent of the node is a container	Property: Nodes.HorizontalAlign <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { HorizontalAlign=HorizontalAlignment.Left })).Render(); }</pre>	Not applicable
Defines an interface in a UML interface Diagram	Property: Nodes.Interface <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new UMLClassifier() { Classifier = ClassifierShapes.Interface, } })}).Render(); }</pre>	Not applicable
Defines the name, attributes and methods of a Interface. Applicable, if the node is a Interface	Property: Nodes.Interface.Name <pre>@{ Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Name = "Patient" } })).Render(); }</pre>	Not applicable
Defines the collection of attributes	Property: Nodes.Interface.Attributes <pre>@{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Name = "accepted" }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }</pre>	Not applicable
Sets the name of the attribute	Property: Nodes.Interface.Attributes.Name <pre>@{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Name = "accepted" }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }</pre>	Not applicable

Sets the data type of attribute	Property: Nodes.Interface.Attributes.Type @{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Type = "Date" }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }	Not applicable
Defines the visibility of the attribute	Property: Nodes.Interface.Attributes.Scope @{ Collection attribute = new Collection(); attribute.Add(new UMLAttribute() { Scope = Scopes.Protected }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }	Not applicable
Defines the collection of methods of an interface	Property: Nodes.Interface.Methods @{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Scope = Scopes.Protected }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }	Not applicable
Sets the name of the method	Property: Nodes.Interface.Methods.Name @{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Name = "Date" }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }	Not applicable
Defines the arguments of the method	Property: Nodes.Interface.Methods.Arguments @{ Collection parameter = new Collection(); parameter.Add(new UMLParameter() { Name = "Date", Type = "String" }); Collection methods = new Collection(); methods.Add(new UMLMethod() { Parameters = parameter });	Not applicable

	<pre>Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }</pre>	
Defines the name, attributes and methods of a interface. Applicable, if the node is a interface	<p>Property: Nodes.Interface.Methods.Arguments.Name</p> <pre>@{ Collection parameter = new Collection(); parameter.Add(new UMLParameter() { Name = "Date" }); Collection methods = new Collection(); methods.Add(new UMLMethod() { Parameters = parameter }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }</pre>	Not applicable
Sets the type of the argument	<p>Property: Nodes.Interface.Methods.Arguments.Type</p> <pre>@{ Collection parameter = new Collection(); parameter.Add(new UMLParameter() { Type = "String" }); Collection methods = new Collection(); methods.Add(new UMLMethod() { Parameters = parameter }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }</pre>	Not applicable
Sets the return type of the method	<p>Property: Nodes.Interface.Methods.Type</p> <pre>@{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Type = "History" }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier = Interface, Interface = { Attributes = attribute } })).Render(); }</pre>	Not applicable
Sets the visibility of the method	<p>Property: Nodes.Interface.Methods.Scope</p> <pre>@{ Collection methods = new Collection(); methods.Add(new UMLMethod() { Scope = Scopes.Protected }); Html.EJ().Diagram("diagram")Nodes(n => n.Add(new UMLClassifier() { Classifier =</pre>	Not applicable

	Interface, Interface = { Attributes = attribute }))).Render(); }	
Defines whether the sub tree of the node is expanded or collapsed	Property: Nodes.IsExpanded @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { IsExpanded=false }))).Render(); }	Property: Nodes.IsExpanded [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { IsExpanded = false }); ViewBag.Nodes = nodes;
Sets the node as a swimlane	Property: Nodes.IsSwimlane [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); SwimLane swimLane = new SwimLane(); swimLane.IsSwimlane = true; Model.Nodes.Add(swimLane); ViewData["diagramModel"] = Model;	Not applicable
A collection of objects where each object represents a label	Property: Nodes.Labels @{ Collection Labels = new Collection(); Labels.Add(new Label() { Text = "Connector" }); Html.EJ().Diagram("diagram").Height("500px").Width("500px").Nodes(n => n.Add(new Node() { Name = "Patient", OffsetX = 100, OffsetY = 100, Labels=Labels }))).Render(); }	Property: Nodes.Annotations [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Content = "Annotation" }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;
An array of objects where each object represents a lane. Applicable, if the node is a swimlane	Property: Nodes.Lanes [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Collection lanes = new Collection(); lanes.Add(new Lane() { Name = "lane1" }); lanes.Add(new Lane() { Name = "lane2" }); SwimLane swimLane = new SwimLane(); swimLane.Lanes = lanes;	Not applicable

	<pre>Model.Nodes.Add(swimLane); ViewData["diagramModel"] = Model;</pre>	
<p>This property allows you to customize lanes appearance using user-defined CSS</p>	<p>Property: Nodes.Lanes.CssClass</p> <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] DiagramProperties Model = new DiagramProperties(); Collection lanes = new Collection(); lanes.Add(new Lane() { CssClass = "hoverLane" }); SwimLane swimLane = new SwimLane(); swimLane.Lanes = lanes; Model.Nodes.Add(swimLane); ViewData["diagramModel"] = Model;</pre>	Not applicable
<p>Defines the header of the lane</p>	<p>Property: Nodes.Lanes.Header</p> <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Collection lanes = new Collection(); Header header = new Header(); header.Text = "Text"; lanes.Add(new Lane() { Header = header }); SwimLane swimLane = new SwimLane(); swimLane.Lanes = lanes; swimLane.IsSwimlane = true; Model.Nodes.Add(swimLane);</pre>	Not applicable
<p>Defines the width of lane</p>	<p>Property: Nodes.Lanes.Width</p> <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Collection lanes = new Collection(); lanes.Add(new Lane() { Width = 200 }); SwimLane swimLane = new SwimLane(); swimLane.Lanes = lanes; swimLane.IsSwimlane = true; Model.Nodes.Add(swimLane);</pre>	Not applicable
<p>An array of objects where each object represents a child node of the lane</p>	<p>Property: Nodes.Lanes.Children</p> <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Collection lanes = new Collection(); Collection children = new Collection(); children.Add(new Node() { Name = "process" }); lanes.Add(new Lane() { Children = children }); SwimLane swimLane = new SwimLane(); swimLane.Lanes</pre>	Not applicable

	<code>= lanes; swimLane.IsSwimlane = true; Model.Nodes.Add(swimLane);</code>	
Defines the object as a lane	Property: <code>Nodes.Lanes.IsLane</code> <pre>[View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Collection lanes = new Collection(); lanes.Add(new Lane() { IsLane = true }); SwimLane swimLane = new SwimLane(); swimLane.Lanes = lanes; Model.Nodes.Add(swimLane);</pre>	Not applicable
Defines the minimum space to be left between the bottom of parent bounds and the node	Property: <code>Nodes.Margin</code> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { MarginBottom=50, MarginLeft=10, MarginRight=10, MarginTop=10 })).Render(); }</pre>	Property: <code>Nodes.Margin</code> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Margin = new DiagramMargin() { Bottom = 15, Left = 15, Right = 15, Up = 15 } }); ViewBag.Nodes = nodes;</pre>
Defines the maximum height limit of the node	Property: <code>Nodes.MaxHeight</code> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() {MaxHeight=100})).Render(); }</pre>	Property: <code>Nodes.MaxHeight</code> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { MaxHeight = 50 }); ViewBag.Nodes = nodes;</pre>
Sets the unique identifier of the node	Property: <code>Nodes.Name</code> <pre>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Name = "Patient" })).Render(); }</pre>	Property: <code>Nodes.Id</code> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { ID = "Name" }); ViewBag.Nodes = nodes;</pre>
Sets the path geometry that defines the shape of a path node	Property: <code>Nodes.PathData</code> <pre>@{ Html.EJ().Diagram("diagram").Height("500px"). Width("500px").Nodes(n => n.Add(new BasicShape() { Shape = BasicShapes.Path,</pre>	Property: <code>Nodes.Shape.Data</code> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new</pre>

	PathData= "M 370.9702,194.9961 L 359.5112,159.7291 L 389.5112,137.9341 L 419.5112,159.7291 L 408.0522,194.9961 L 370.9702,194.9961 z"));Render(); }	DiagramNode() { Shape = new DiagramBasicShape() { Type = Shapes.Path } }); ViewBag.Nodes = nodes;
An array of objects, where each object represents a smaller region(phase) of a swimlane	Property: Nodes.Phases [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] Collection phase = new Collection(); phase.Add(new Phase() { Name = "Phase1" }); SwimLane swimLane = new SwimLane(); swimLane.Phases = phase; Model.Nodes.Add(swimLane);	Not applicable
Defines the header of the smaller regions	Property: Nodes.Phases.Label [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] Collection label = new Collection(); label.Add(new Label() { Text = "Phase1" }); Collection phase = new Collection(); phase.Add(new Phase() { Labels = label }); SwimLane swimLane = new SwimLane(); swimLane.Phases = phase; Model.Nodes.Add(swimLane);	Not applicable
Defines the line color of the splitter that splits adjacent phases	Property: Nodes.Phases.LineColor [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] Collection phase = new Collection(); phase.Add(new Phase() { LineColor = "green" }); SwimLane swimLane = new SwimLane(); swimLane.Phases = phase; Model.Nodes.Add(swimLane);	Not applicable
Sets the length of the smaller region(phase) of a swimlane	Property: Nodes.Phases.Offset [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel "] as DiagramProperties).Render()[Model] Collection phase = new Collection(); phase.Add(new Phase() { Offset = "150" }); SwimLane swimLane = new SwimLane();	Not applicable

	swimLane.Phases = phase; Model.Nodes.Add(swimLane);	
Sets the orientation of the phase	Property: Nodes.Phases.Orientation [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Collection phase = new Collection(); phase.Add(new Phase() { Orientation = "Horizontal" }); SwimLane swimLane = new SwimLane(); swimLane.Phases = phase; Model.Nodes.Add(swimLane);	Not applicable
Sets the height of the phase headers	Property: Nodes.PhaseSize [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Collection phase = new Collection(); phase.Add(new Phase() { Orientation = "Horizontal" }); SwimLane swimLane = new SwimLane(); swimLane.PhaseSize = "50"; Model.Nodes.Add(swimLane);	Not applicable
Sets the ratio/fractional value relative to node, based on which the node will be transformed (positioning, scaling and rotation)	Property: Nodes.Pivot @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Pivot = new DiagramPoint() { X=0, Y=0 } })).Render(); }	Property: Nodes.Pivot [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Pivot = new DiagramPoint() { X = 0, Y = 0 } }); ViewBag.Nodes = nodes;
Defines a collection of points to draw a polygon. Applicable, if the shape is a polygon	Property: Nodes.Points @{ Collection point = new Collection(); point.Add(new DiagramPoint() { X = 0, Y = 12.5F }); point.Add(new DiagramPoint() { X = 0, Y = 50 }); point.Add(new DiagramPoint() { X = 50, Y = 50 }); point.Add(new DiagramPoint() { X = 50, Y = 0 }); point.Add(new DiagramPoint() { X = 12.5F, Y = 0 }); point.Add(new DiagramPoint() { X = 0, Y = 12.5F }); Html.EJ().Diagram("diagram").Height("500px").	Property: Nodes.Shape.Points [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List points = new List(); points.Add(new DiagramPoint() { X = 35, Y = 0 }); points.Add(new DiagramPoint() { X = 65, Y = 0 }); points.Add(new DiagramPoint() { X = 100, Y = 35 }); points.Add(new DiagramPoint() { X = 100, Y = 65 });

	<pre>Width("500px").Nodes(n => n.Add(new BasicShape() { Name = "Patient", OffsetX = 100, OffsetY = 100, Shape=Syncfusion.JavaScript.DataVisualization .DiagramEnums.BasicShapes.Polygon, Points= point })).Render(); }</pre>	<pre>points.Add(new DiagramPoint() { X = 65, Y = 100 }); points.Add(new DiagramPoint() { X = 35, Y = 100 }); points.Add(new DiagramPoint() { X = 0, Y = 65 }); points.Add(new DiagramPoint() { X = 0, Y = 35 }); List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new DiagramBasicShape() { Type = Basic, Shape = BasicShapes.Polygon, Points = points } }); ViewBag.Nodes = nodes;</pre>
An array of objects where each object represents a port	<p>Property: Nodes.Ports</p> <pre>@{ Collection ports = new Collection(); ports.Add(new Port() { Name="port1", Offset=new DiagramPoint() {X=0.5F, Y=0 }, Shape=PortShapes.Square, Size=10 }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Ports=ports })).Render(); }</pre>	<p>Property: Nodes.Ports</p> <p>[View]</p> <pre>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List ports = new List(); ports.Add(new DiagramPort() { Id = "port" }); List nodes = new List(); nodes.Add(new DiagramNode() { Ports = ports }); ViewBag.Nodes = nodes;</pre>
Sets the border color of the port	<p>Property: Nodes.Ports.BorderColor</p> <pre>@{ Collection ports = new Collection(); ports.Add(new Port() { BorderColor= "Yellow" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Ports=ports })).Render(); }</pre>	<p>Property: Nodes.Ports.Style.StrokeColor</p> <p>[View]</p> <pre>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List ports = new List(); ports.Add(new DiagramPort() { }); List nodes = new List(); nodes.Add(new DiagramNode() { Ports = ports }); ViewBag.Nodes = nodes;</pre>
Defines whether connections can be created with the port	<p>Property: Nodes.Ports.Constraints</p> <pre>@{ Collection ports = new Collection(); ports.Add(new Port() { Constraints=PortConstraints.Connect }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Ports=ports })).Render(); }</pre>	<p>Property: Nodes.Ports.Constraints</p> <p>[View]</p> <pre>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List ports = new List(); ports.Add(new DiagramPort() { Constraints = PortConstraints.Drag }); List nodes = new List(); nodes.Add(new DiagramNode() { Ports = ports }); ViewBag.Nodes = nodes;</pre>
An array of objects where each object	<p>Property: Nodes.Ports.Shape</p> <pre>@{ Collection ports = new Collection(); ports.Add(new Port() { Shape=PortShapes.Square });</pre>	<p>Property: Nodes.Ports.Shape</p> <p>[View]</p> <pre>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List ports = new List(); ports.Add(new</pre>

represents a port	<code>Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Ports=ports })).Render(); }</code>	<code>DiagramPort() { Shape = PortShapes.Circle }; List nodes = new List(); nodes.Add(new DiagramNode() { Ports = ports }); ViewBag.Nodes = nodes;</code>
Sets the vertical alignment of the port with respect to its immediate parent	Not applicable	Property: <code>Nodes.Ports.VerticalAlignment</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List ports = new List(); ports.Add(new DiagramPort() { VerticalAlignment = VerticalAlignment.Top }); List nodes = new List(); nodes.Add(new DiagramNode() { Ports = ports }); ViewBag.Nodes = nodes;</code>
Defines the opacity and the position of shadow	Property: <code>Nodes.Shadow</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Shadow=new Shadow() { Opacity=0.5F } })).Render(); }</code>	Property: <code>Nodes.Shadow</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shadow = new DiagramShadow() { Opacity = 0.5 } }); ViewBag.Nodes = nodes;</code>
Defines the sub process of a BPMN Activity. Applicable, if the type of the BPMN activity is sub process	Property: <code>Nodes.SubProcess</code> <code>@{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape=BPMNShapes.Activity, Activity=BPMNActivity.SubProcess, SubProcess = { Loop=BPMNLoops.Standard, Adhoc=true } })).Render(); }</code>	Property: <code>Nodes.Shape.Activity.SubProcess</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Activity", Activity = { Activity = BpmnActivities.SubProcess } }); ViewBag.Nodes = nodes;</code>
Defines the collection of events that need to be appended with BPMN Sub-Process	Property: <code>Nodes.SubProcess.Events</code> <code>@{ Collection events = new Collection(); events.Add(new BPMNEvent() { Name = "intermediate1" } }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.Activity, Activity = BPMNActivity.SubProcess, SubProcess = { Type</code>	Property: <code>Nodes.Shape.Activity.SubProcess.Events</code> [View] <code>@Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List events = new List(); events.Add(new DiagramBpmnSubEvent() { Event =</code>

	<pre>= BPMNSubProcessTypes.Event, Events=events } })).Render(); }</pre>	<pre>BpmnEvents.Intermediate }); List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Activity", Activity = { Activity = BpmnActivities.SubProcess, SubProcess = { Type = BpmnSubProcessTypes.Event, Events = events } } } }); ViewBag.Nodes = nodes;</pre>
An array of objects where each object represents a port	<pre>Property:Nodes.SubProcess.Events.Ports</pre> <pre>@{ Collection ports = new Collection(); ports.Add(new Port() { Name = "port1" }); Collection events = new Collection(); events.Add(new BPMNEvent() { Ports = ports } }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.Activity, Activity = BPMNActivity.SubProcess, SubProcess = { Type = BPMNSubProcessTypes.Event, Events=events } })).Render(); }</pre>	<pre>Property:Nodes.Shape.Activity.SubProcess.Events.Ports</pre> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List ports = new List(); ports.Add(new DiagramPort() { Id = "port" }); List events = new List(); events.Add(new DiagramBpmnSubEvent() { Event = BpmnEvents.Intermediate, Ports = ports }); List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Activity", Activity = { Activity = BpmnActivities.SubProcess, SubProcess = { Type = BpmnSubProcessTypes.Event, Events = events } } } }); ViewBag.Nodes = nodes;</pre>
A collection of objects where each object represents a label	<pre>Property:Nodes.SubProcess.Events.Labels</pre> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Text = "Label" }); Collection events = new Collection(); events.Add(new BPMNEvent() { Labels = Labels } }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.Activity, Activity = BPMNActivity.SubProcess, SubProcess = { Type = BPMNSubProcessTypes.Event, Events=events } })).Render(); }</pre>	<pre>Property:Nodes.Shape.Activity.SubProcess.Events.Annotations</pre> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Content = "Annotation" }); List events = new List(); events.Add(new DiagramBpmnSubEvent() { Event = BpmnEvents.Intermediate, Annotation = annotation }); List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Activity", Activity = { Activity = BpmnActivities.SubProcess, SubProcess =</pre>

		{ Type = BpmnSubProcessTypes.Event, Events = events } } } }); ViewBag.Nodes = nodes;
Defines the task of the BPMN activity. Applicable, if the type of activity is set as task	Property: Nodes.Task @{ Html.EJ().Diagram("diagram").Nodes(n => n.Add(new BPMNNode() { Shape = BPMNShapes.Activity, Activity = BPMNActivity.Task, Task = { Type=BPMNTasks.Service } })).Render(); }	Property: Nodes.Shape.Activity.Task [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List nodes = new List(); nodes.Add(new DiagramNode() { Shape = new BpmnShapes() { Type = "Bpmn", Shape = "Activity", Activity = BpmnActivities.Task, Task = { Type = BpmnTasks.Service } } } }); ViewBag.Nodes = nodes;

NodeTemplate

behavior	API in Essential JS 1	API in Essential JS 2
Binds the custom JSON data with node properties	Property: NodeTemplate @{ Collection Labels = new Collection(); Labels.Add(new Label() { Text = "Label" } }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }	Property: SetNodeTemplate @Html.EJS().Diagram("container").SetNodeTemplate("setNodeTemplate").Render()

Layers

behavior	API in Essential JS 1	API in Essential JS 2
A collection of JSON objects where each object represents a layer. Layer is a named	Property: Layers [View] @Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Model.Layers = new Collection(); Model.Layers.Add(new Layer() { Name = "Layer1" });	Property: Layers [View] @Html.EJS().Diagram("container").Layers(ViewBag.Layers).Render()[Model] List layers = new List(); layers.Add(new DiagramLayer() { Id = "layer" }); ViewBag.Layers = layers;

category of diagram shapes		
A collection of JSON objects where each object represents a layer. Layer is a named category of diagram shapes	Property: Layers.Print [View] @Html.EJ().Diagram("Diagram 1", ViewData["diagramModel"] as DiagramProperties).Render()[Model] Model.Layers = new Collection(); Model.Layers.Add(new Layer() { Print = true });	Not applicable

Annotations

behavior	API in Essential JS 1	API in Essential JS 2
A collection of objects where each object represents a label	Property: Labels.Text @{ Collection Labels = new Collection(); Labels.Add(new Label() { Text = "Label" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }	Property: Annotations.Content [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Content = "Annotation" }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;
Offset for annotation content	Property: Labels.Offset @{ Collection Labels = new Collection(); Labels.Add(new Label() { Offset=new DiagramPoint() { X = 0, Y = 1 } }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }	Property: Annotations.Offset [View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Offset=new DiagramPoint() { X = 0, Y = 1 } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;
Sets the hyperlink	Property: Labels.HyperLink	Property: Annotations.Hyperlink

for the labels	<pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Text = "HRPortal", Hyperlink="https://www.Syncfusion.Com" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Content = "HRPortal", Hyperlink = "https://www.Syncfusion.Com" }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Enables/disables the bold style	<p>Property: Labels.Bold</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Bold=true }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.Style.Bold</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { Bold = true } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the border color of the label	<p>Property: Labels.BorderColor</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { BorderColor="red" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.Style.StrokeColor</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { StrokeColor = "red" } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the border width of the label	<p>Property: Labels.BorderWidth</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { BorderWidth=2 }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.Style.StrokeWidth</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { StrokeWidth=2 } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
This property allows you to customize labels appearance using user-defined CSS	<p>Property: Labels.CssClass</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { CssClass= "hoverText" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Not applicable

Enables or disables the default behaviors of the label	Property: Labels.Constraints <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Constraints = LabelConstraints.Resizable }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Property: Annotations.Constraints <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Constraints = AnnotationConstraints.InheritReadOnly }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the fill color of the text area	Property: Labels.FillColor <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { FillColor= "green" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Property: Annotations.Style.Fill <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { Fill = "white" } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the font color of the text	Property: Labels.FontColor <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { FontColor="blue" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Property: Annotations.Style.Color <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { Color = "black" } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the font family of the text	Property: Labels.FontFamily <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { FontFamily= "segoe UI" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Property: Annotations.Style.FontFamily <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { FontFamily = "segoe UI" } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the height of the label	Property: Labels.Height <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Height = 100 }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Property: Annotations.Height <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Height = 10 }); List nodes = new List(); nodes.Add(new</pre>

	<pre>es(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<pre>DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the horizontal alignment of the label	<p>Property: Labels.HorizontalAlignment</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { HorizontalAlignment=HorizontalAlignment.Right }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.HorizontalAlignment</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { HorizontalAlignment = HorizontalAlignment.Center }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
To set the margin of the label	<p>Property: Labels.Margin</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Margin=new LabelMargin() { Left=5 } }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.Margin</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Margin = new DiagramMargin() { Bottom = 15 } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Defines whether the label is editable or not	<p>Property: Labels.ReadOnly</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { ReadOnly=true }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.Constraints</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Constraints = AnnotationConstraints.ReadOnly }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Sets the id of svg/html templates. Applicable, if the node's label is HTML or native	<p>Property: Labels.TemplateId</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { TemplateId= "SvgEllipse" }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Not applicable
Defines how to align the	<p>Property: Labels.TextAlign</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new</pre>	<p>Property: Annotations.Style.TextAlign</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes)</pre>

text inside the label	<pre>Label() { TextAlign=TextAlign.Left }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<pre>es).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Style = new DiagramTextStyle() { TextAlign = TextAlign.Left } }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Enables or disables the visibility of the label	<p>Property: Labels.Visible</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Visible=false }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	<p>Property: Annotations.Visibility</p> <pre>[View] @Html.EJS().Diagram("container").Nodes(ViewBag.Nodes).Render()[Model] List annotation = new List(); annotation.Add(new DiagramNodeAnnotation() { Visibility = false }); List nodes = new List(); nodes.Add(new DiagramNode() { Annotations = annotation }); ViewBag.Nodes = nodes;</pre>
Gets whether the label is currently being edited or not	<p>Property: Labels.Mode</p> <pre>@{ Collection Labels = new Collection(); Labels.Add(new Label() { Mode = LabelEditMode.View }); Html.EJ().Diagram("diagram").Nodes(n => n.Add(new Node() { Labels = Labels })).Render(); }</pre>	Not applicable

PageSettings

behavior	API in Essential JS 1	API in Essential JS 2
Defines the size and appearance of diagram page	<p>Property: PageSettings.AutoScrollBorder</p> <pre>@{ Html.EJ().Diagram("diagram").PageSettings(p=>p.AutoScrollBorder(new AutoScrollBorder() { Left=50, Top=50, Bottom=50, Right=50})).Render(); }</pre>	Not applicable
Sets whether multiple pages can be created to fit all nodes and	<p>Property: PageSettings.MultiplePage</p> <pre>@{ Html.EJ().Diagram("diagram").PageSettings(p=>p.MultiplePage(false)).Render(); }</pre>	<p>Property: PageSettings.MultiplePage</p> <pre>@Html.EJS().Diagram("container").PageSettings(new DiagramPageSettings() { MultiplePage= true }).Render()</pre>

connectors		
Defines the background color of diagram pages	Property: PageSettings.PageBackgroundColor <pre>@{ Html.EJ().Diagram("diagram").PageSettings(p=>p.PageBackground("Grey")).Render(); }</pre>	Property: PageSettings.Background.Color <pre>@Html.EJS().Diagram("container").PageSettings(new DiagramPageSettings() { Background = new DiagramBackground() { Color="red" } }).Render()</pre>
Defines the scrollable area of diagram. Applicable, if the scroll limit is not limited.	Property: PageSettings.ScrollableArea <pre>@{ Html.EJ().Diagram("diagram").PageSettings(p=>p.ScrollableArea(new ScrollableArea() { Height=300, Width=300, X=0, Y= 0})).Render(); }</pre>	Property: ScrollSettings.ScrollableArea <pre>@Html.EJS().Diagram("container").ScrollSettings(new DiagramScrollSettings() { ScrollableArea = "getScrollableArea" }).Render()</pre>
Defines the draggable region of diagram elements	Property: PageSettings.BoundaryConstraints <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.PageSettings = new PageSettings() { BoundaryConstraints = BoundaryConstraints.Diagram }; ViewData["diagramModel"] = Model;</pre>	Property: PageSettings.BoundaryConstraints <pre>@Html.EJS().Diagram("container").PageSettings(new DiagramPageSettings() { BoundaryConstraints= BoundaryConstraints.Diagram }).Render()</pre>

ScrollSettings

behavior	API in Essential JS 1	API in Essential JS 2
Defines the zoom value, zoom factor, scroll status and view port size of the diagram	Property: ScrollSettings.HorizontalOffset <pre>[View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.ScrollSettings = new</pre>	Property: ScrollSettings.HorizontalOffset <pre>@Html.EJS().Diagram("container").ScrollSettings(new DiagramScrollSettings() { HorizontalOffset = 300 }).Render()</pre>

	ScrollSettings() { HorizontalOffset = 300 }; ViewData["diagramModel"] = Model;	
Allows to extend the scrollable region that is based on the scroll limit	Property: ScrollSettings.Padding [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties);[Model] DiagramProperties Model = new DiagramProperties(); Model.ScrollSettings = new ScrollSettings() { Padding = { Left = 25, Bottom = 25, Right = 25, Top = 25 } }; ViewData["diagramModel"] = Model;	Not applicable
Allows to read the maximum zoom value of scroller	Not applicable	Property: ScrollSettings.MaxZoom @Html.EJS().Diagram("container").ScrollSettings (new DiagramScrollSettings() { MaxZoom = 2.5 }).Render()
Allows to read the maximum zoom value of scroller	Not applicable	Property: ScrollSettings.AutoScrollBorder @Html.EJS().Diagram("container").ScrollSettings (new DiagramScrollSettings() { MaxZoom = 2.5 }).Render()
Enables/Disables the autoscroll option	Not applicable	Property: ScrollSettings.CanAutoScroll @Html.EJS().Diagram("container").ScrollSettings (new DiagramScrollSettings() { CanAutoScroll = true }).Render()
Defines the scrollable area of diagram	Not applicable	Property: ScrollSettings.ScrollableArea @Html.EJS().Diagram("container").ScrollSettings (new DiagramScrollSettings() { ScrollableArea = "getScrollableArea" }).Render()

SnapSettings

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables snappin	Property: SnapSettings.EnableSnapToObject @{	Not applicable

g nodes/c onnecto rs to objects	Html.EJ().Diagram("diagram").SnapSettings(s=>s.EnableSnapToObject(true)).Render(); }	
Defines the appear ance of horizont al gridlines	Property: SnapSettings.HorizontalGridLines @{ decimal lineInterval = { 1, 14, 0.5F, 14.5F}; Html.EJ().Diagram("diagram").SnapSettings(s=>s.HorizontalGridLines(h=>h.LinesInterval(lineInterval))).Render(); }	Property: SnapSettings.HorizontalGridlines [View] @Html.EJS().Diagram("container").SnapSettings(ViewBag.SnapSettings).Render()[Model] double[] intervals = { 0.95, 9.05, 0.2, 9.75 }; DiagramSnapSettings snapSettings = new DiagramSnapSettings(); snapSettings.HorizontalGridlines = new DiagramGridlines() { LineIntervals = intervals }; ViewBag.SnapSettings = snapSettings;
Defines the appear ance of vertical gridlines	Property: SnapSettings.VerticalGridLines @{ decimal lineInterval = { 1, 14, 0.5F, 14.5F}; Html.EJ().Diagram("diagram").SnapSettings(s=>s.VerticalGridLines(v=> v.LinesInterval(lineInterval))).Render(); }	Property: SnapSettings.VerticalGridLines [View] @Html.EJS().Diagram("container").SnapSettings(ViewBag.SnapSettings).Render()[Model] double[] intervals = { 0.95, 9.05, 0.2, 9.75 }; DiagramSnapSettings snapSettings = new DiagramSnapSettings(); snapSettings.VerticalGridLines = new DiagramGridlines() { LineIntervals = intervals }; ViewBag.SnapSettings = snapSettings;
Defines the angle by which the object needs to be snapped	Property: SnapSettings.SnapAngle @{ Html.EJ().Diagram("diagram").SnapSettings(s=>s.SnapAngle(5)).Render(); }	Property: SnapSettings.SnapAngle [View] @Html.EJS().Diagram("container").SnapSettings(ViewBag.SnapSettings).Render()[Model] DiagramSnapSettings snapSettings = new DiagramSnapSettings(); snapSettings.SnapAngle = 5; ViewBag.SnapSettings = snapSettings;

Sets the minimum distance between the selected object and the nearest object	Property: SnapSettings.SnapObjectDistance <pre>@{ Html.EJ().Diagram("diagram").SnapSettings(s=>s.SnapObjectDistance(5)).Render(); }</pre>	Property: SnapSettings.SnapObjectDistance <pre>[View] @Html.EJS().Diagram("container").SnapSettings(ViewBag.SnapSettings).Render([Model] DiagramSnapSettings snapSettings = new DiagramSnapSettings(); snapSettings.SnapObjectDistance = 5; ViewBag.SnapSettings = snapSettings;</pre>
Defines and sets the snapConstraints	Property: SnapSettings.SnapConstraints <pre>@{ Html.EJ().Diagram("diagram").SnapSettings(s=>s.SnapConstraints(SnapConstraints.ShowLines)).Render(); }</pre>	Property: SnapSettings.Constraints <pre>[View] @Html.EJS().Diagram("container").SnapSettings(ViewBag.SnapSettings).Render([Model] DiagramSnapSettings snapSettings = new DiagramSnapSettings(); snapSettings.Constraints = SnapConstraints.ShowLines; ViewBag.SnapSettings = snapSettings;</pre>

ZoomFactor

behavior	API in Essential JS 1	API in Essential JS 2
Sets the factor by which we can zoom in or zoom out	Property: ZoomFactor <pre>@{ Html.EJ().Diagram("diagram").ZoomFactor(1).Render(); }</pre>	Property: ZoomFactor <pre>var zoomIn = { type: 'ZoomIn', zoomFactor: 0.5 }; diagram.ZoomTo(zoomIn);</pre>

Tool

behavior	API in Essential JS 1	API in Essential JS 2
Enables/Disables the interactive behaviors of diagram	Property: Tool <pre>@{ Html.EJ().Diagram("diagram").Tool(Tool.ZoomPan).Render(); }</pre>	Property: Tool <pre>@Html.EJS().Diagram("container").Tool(DiagramTools.ZoomPan).Render()</pre>

ShowTooltip

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables tooltip of diagram	Property: <code>ShowTooltip</code> <pre>@{ Html.EJ().Diagram("diagram").Height("500px").Width("500px").ShowTooltip(true).Render(); }</pre>	Property: <code>Constraints</code> <pre>@Html.EJS().Diagram("container").Constraints(DiagramConstraints.Default DiagramConstraints.Tooltip).Render()</pre>

SelectedItems

behavior	API in Essential JS 1	API in Essential JS 2
A read only collection of the selected items	Property: <code>SelectedItems.Children</code> <pre>var diagram = \$("#diagramcontent").EjDiagram("instance"); for(var i=0; i< diagram.Model.SelectedItems.Children; i++){ //Do your actions here }</pre>	Not applicable
Controls the visibility of selector	Property: <code>SelectedItems.Constraints</code> <pre>@{ Html.EJ().Diagram("diagram").SelectedItems(new SelectedItems() { Constraints = SelectorConstraints.UserHandles } } }).Render(); }</pre>	Property: <code>SelectedItems.Constraints</code> <pre>@Html.EJS().Diagram("container").SelectedItems(new DiagramSelector() { Constraints = SelectorConstraints.UserHandle}).Render()</pre>
Sets the angle to rotate the selected items	Property: <code>SelectedItems.Tooltip</code> <pre>@{ Html.EJ().Diagram("diagram").SelectedItems(new SelectedItems() { Tooltip = new Tooltip() { Alignment = { Vertical = VerticalAlignment.Top } } }).Render(); }</pre>	Not applicable
A collection of	Property: <code>SelectedItems.UserHandles</code> <pre>var userHandle= []; var cloneHandle =</pre>	Property: <code>SelectedItems.UserHandles</code> <pre>[View]</pre>

frequency used commands that will be added around the selector	<pre>ej.Datavisualization.Diagram.UserHandle(); userHandle.Push(cloneHandle); var diagram = \$("#DiagramContent").EjDiagram("instance"); \$("#diagramcontent").EjDiagram({ selectedItems: { userHandles:userHandle } });</pre>	<pre>@Html.EJS().Diagram("container").SelectedItems(ViewBag.Selector).Render()[Model] List userHandle = new List(); userHandle.Add(new DiagramUserHandle() { Name = "clone", PathData = 'M 60.3,18 H 27.5 c -3,0-5.5,2.4-5.5,5.5 v 38.2 h 5.5 V 23.5 h 32.7 V 18 z M 68.5,28.9 h -30 c -3,0-5.5,2.4-5.5,5.5 v 38.2 c 0,3,2.4,5.5,5.5,5.5 h 30 c 3,0,5.5-2.4,5.5-5.5 V 34.4 C 73.9,31.4,71.5,28.9,68.5,28.9 z M 68.5,72.5 h -30 V 34.4 h 30 V 72.5 z', }); DiagramSelector selector = new DiagramSelector(); selector.UserHandles = userHandle; ViewBag.Selector = selector;</pre>
Sets the horizontal alignment of the user handle	<pre>Property:SelectedItems.UserHandles.HorizontalAlignment var userHandle = []; var cloneHandle = ej.Datavisualization.Diagram.UserHandle(); cloneHandle = {name : "cloneHandle", pathData : "M 4.6350084,4.8909971 L 4.6350084,9.3649971 9.5480137,9.3649971 9.5480137,4.8909971 z M 3.0000062,2.8189973 L 11.184016,2.8189973 11.184016,10.999997 3.0000062,10.999997 z M 0,0 L 7.3649998,0 7.3649998,1.4020001 1.4029988,1.4020001 1.4029988,8.0660002 0,8.0660002 0,1.4020001 0,0.70300276 z", visible : "true", backgroundColor : "#4D4D4D", offset : ej.Datavisualization.Diagram.Point(0, 0), position : "middleLeft" margin : { left: 5 }, pathColor : "white", horizontalAlignment : ej.Datavisualization.Diagram.HorizontalAlignment.Right, verticalAlignment : ej.Datavisualization.Diagram.VerticalAlignment.Top, borderColor : "red", borderWidth : 3, size : 20}; userHandle.Push(cloneHandle); \$("#diagramcontent").EjDiagram({ selectedItems: { userHandles:userHandle } });</pre>	<pre>Property:SelectedItems.UserHandles.HorizontalAlignment [View] @Html.EJS().Diagram("container").SelectedItems(ViewBag.Selector).Render()[Model] List userHandle = new List(); userHandle.Add(new DiagramUserHandle() { HorizontalAlignment = HorizontalAlignment.Auto }); DiagramSelector selector = new DiagramSelector(); selector.UserHandles = userHandle; ViewBag.Selector = selector;</pre>

Define the interactive behaviors of the user handle	Property: SelectedItems.UserHandles. Tool <pre> var CloneTool = (function(base) { ej.Datavisualization.Diagram.Extend(Clo neTool, base); function CloneTool(name) { base.Call(this, name); this.SingleAction = true; this.ClonedNodes = []; this.Cursor = "pointer"; } CloneTool.Prototype.Mouseup = function(event) { this.Diagram.Copy(); this.Diagram.Paste(); } } return CloneTool; }); (ej.Datavisualization.Diagram.ToolBase); var userHandle = []; var cloneHandle = ej.Datavisualization.Diagram.UserHandle (); cloneHandle.Name = "cloneHandle"; cloneHandle.PathData = "M 4.6350084,4.8909971 L 4.6350084,9.3649971 9.5480137,9.3649971 9.5480137,4.8909971 z M 3.0000062,2.8189973 L 11.184016,2.8189973 11.184016,10.999997 3.0000062,10.999997 z M 0,0 L 7.3649998,0 7.3649998,1.4020001 1.4029988,1.4020001 1.4029988,8.0660002 0,8.0660002 0,1.4020001 0,0.70300276 z"; cloneHandle.Tool = new CloneTool(cloneHandle.Name);; userHandle.Push(cloneHandle); \$("#diagramcontent").EjDiagram({ selectedItems: { userHandles: userHandle } }); </pre>	Not applicable
Define whether the user handle should be added, when more	Property: SelectedItems.UserHandles. EnableMultiSelection <pre> var userHandle = []; var cloneHandle = ej.Datavisualization.Diagram.UserHandle (); cloneHandle.Name = "cloneHandle"; cloneHandle.EnableMultiSelection = true; userHandle.Push(cloneHandle); \$("#diagramcontent").EjDiagram({ </pre>	Not applicable

than one element is selected	selectedItems: { userHandles: userHandle } });	
Sets the horizontal alignment of the user handle	Not applicable	Property: SelectedItems.UserHandles.Displacement [View] @Html.EJS().Diagram("container").SelectedItems(ViewBag.Selector).Render()[Model] List userHandle = new List(); userHandle.Add(new DiagramUserHandle() { Displacement = 30 }); DiagramSelector selector = new DiagramSelector(); selector.UserHandles = userHandle; ViewBag.Selector = selector;

SerializationSettings

behavior	API in Essential JS 1	API in Essential JS 2
Defines whether the default diagram properties can be serialized or not	Property: SerializationSettings.PreventDefaultValues [View] @Html.EJ().Diagram("diagram", ViewData["diagramModel"] as DiagramProperties); [Model] DiagramProperties Model = new DiagramProperties(); Model.SerializationSettings = new SerializationSettings() { PreventDefaultValues = true }; ViewData["diagramModel"] = Model;	Not applicable

How to load EJ1 diagram in EJ2 diagram

To load EJ1 JSON data in an EJ2 diagram, follow these steps.

1. Import and inject the EJ1SerializationModule as shown in the following code example.

HTML

```
@{
    Html.EJ().Diagram("diagram").Height("500px").Width("500px").Created("diagramCreated").GetNodeDefaults("getNodeDefaults").GetConnectorDefaults("getConnectorDefaults").DataSourceSettings(ss =>
        ss.Id("Id").ParentId("ParentId").DataSource(new DataManager() { Data =
            (List<MindMapDetails>)ViewBag.Nodes })
    ).Layout(l =>
        l.Type(Syncfusion.EJ2.Diagrams.LayoutType.MindMap).GetBranch("getBranch").Render();
    }
}
```

2. Load the EJ1 JSON data using the diagram loadDiagram method and set the second parameter to true.

HTML

```
function diagramCreated() {
  var diagram = document.getElementById("diagram").ej2_instances[0];
  var ej1Data = {"JSONData"}; // Replace JSONData with your EJ1 JSON data
  //Load the EJ1 JSON and pass a boolean value as true.
  diagram.loadDiagram(ej1Data, true);
}
```

Tooltip

behavior	API in Essential JS 1	API in Essential JS 2
An object that defines the description, appearance and alignments of tooltip	Property: Tooltip @{ Html.EJ().Diagram("diagram").Tooltip(new Diagram.Tooltip() { TemplateId= "mouseOverTooltip" }).Render(); }	Property: Tooltip @Html.EJS().Diagram("container").Constraints(DiagramConstraints.Default DiagramConstraints.Tooltip).Tooltip(new DiagramDiagramTooltip() { Content = "Diagram"}).Render()
Defines the alignment of tooltip	Property: Tooltip.Alignment @{ Html.EJ().Diagram("diagram").Tooltip(new Diagram.Tooltip() { Alignment= new Diagram.Alignment() { Horizontal=HorizontalAlignment.Left })}).Render(); }	Not applicable
Sets the margin of the tooltip	Property: Tooltip.Margin @{ Html.EJ().Diagram("diagram").Tooltip(new Diagram.Tooltip() { Margin= new Margin() { Left=5, Bottom=5, Right=5, Top=5 }).Render(); }	Not applicable
Sets the svg/html template to be	Property: Tooltip.TemplateId @{ Html.EJ().Diagram("diagram").To	Property: Tooltip.Content @Html.EJS().Diagram("container").Tooltip(new DiagramDiagramTooltip() { Content = "Diagram"}).Render()

bound with tooltip	oltip(new Diagram.Tooltip() { TemplateId= "mouseOverTooltip" }).Render(); }	
Defines if the Tooltip has tip pointer or not	Not applicable	Property: Tooltip.ShowTipPointer @Html.EJS().Diagram("container").Tooltip(new DiagramDiagramTooltip() { ShowTipPointer = true }).Render()
Defines the position of the Tooltip	Not applicable	Property: Tooltip.Position @Html.EJS().Diagram("container").Tooltip(new DiagramDiagramTooltip() { Position = "TopLeft" }).Render()

How To

HTML Template and CSS in the Organization chart

An organizational chart is a diagram that displays the structure of an organization and relationships. To create an organizational chart, the [type](#) of layout should be set as an **OrganizationalChart**.

CUSTOMIZEDORGANIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using Syncfusion.EJ2.Diagrams;
namespace organization.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }
        public IActionResult Index()
        {
            ViewBag.Nodes = OverviewData.GetAllRecords();
            return View();
        }
        public class OverviewData
        {
            public string Id { get; set; }
            public string Name { get; set; }
            public string Designation { get; set; }
            public string ReportingPerson { get; set; }
        }
    }
}
```

```

public string Image { get; set; }
public OverviewData(string id, string name, string designation,
string reportingperson, string Image)
{
    this.Id = id;
    this.Name = name;
    this.Designation = designation;
    this.ReportingPerson = reportingperson;
    this.Image = Image;
}
public static List<OverviewData> GetAllRecords()
{
    List<OverviewData> data = new List<OverviewData>();
    data.Add(new OverviewData("parent", "Maria Anders",
"Managing Director", "", "./assets/diagram/employees/image1.png"));
    data.Add(new OverviewData("1", "Ana Trujillo", "Project
Manager", "parent", "./assets/diagram/employees/image2.png"));
    data.Add(new OverviewData("2", "Anto Moreno", "Project
Lead", "1", "./assets/diagram/employees/image3.png"));
    data.Add(new OverviewData("3", "Thomas Hardy", "Senior S/w
Engg", "2", "./assets/diagram/employees/image4.png"));
    data.Add(new OverviewData("4", "Christina kaff", "S/w Engg",
"3", "./assets/diagram/employees/image5.png"));
    data.Add(new OverviewData("5", "Hanna Moos", "Project
Trainee", "4", "./assets/diagram/employees/image6.png"));
    data.Add(new OverviewData("6", "Peter Citeaux", "S/w Engg",
"5", "./assets/diagram/employees/image7.png"));
    data.Add(new OverviewData("7", "Martín Kloss", "Project
Trainee", "6", "./assets/diagram/employees/image8.png"));
    data.Add(new OverviewData("8", "Elizabeth Mary", "Project
Trainee", "6", "./assets/diagram/employees/image9.png"));
    data.Add(new OverviewData("9", "Victoria Ash", "Senior S/w
Engg", "5", "./assets/diagram/employees/image10.png"));
    data.Add(new OverviewData("10", "Francisco Yang", "Senior
S/w Engg", "3", "./assets/diagram/employees/image11.png"));
    data.Add(new OverviewData("11", "Yang Wang", "Project
Manager", "parent", "./assets/diagram/employees/image12.png"));
    data.Add(new OverviewData("12", "Lino Rodri", "Project
Manager", "11", "./assets/diagram/employees/image13.png"));
    data.Add(new OverviewData("13", "Philip Cramer", "Senior S/w
Engg", "24", "./assets/diagram/employees/image14.png"));
    data.Add(new OverviewData("14", "Pedro Afonso", "Project
Trainee", "15", "./assets/diagram/employees/image15.png"));
    data.Add(new OverviewData("15", "Elizabeth Roel", "S/w
Engg", "13", "./assets/diagram/employees/image16.png"));
    data.Add(new OverviewData("16", "Janine Labrune", "Project
Lead", "12", "./assets/diagram/employees/image17.png"));
    data.Add(new OverviewData("17", "Ann Devon", "Project
Manager", "25", "./assets/diagram/employees/image18.png"));
    data.Add(new OverviewData("18", "Roland Mendel", "Project
Lead", "17", "./assets/diagram/employees/image19.png"));
    data.Add(new OverviewData("19", "Aria Cruz", "Senior S/w
Engg", "18", "./assets/diagram/employees/image20.png"));
    data.Add(new OverviewData("20", "Martine Rancé", "S/w Engg",
"18", "./assets/diagram/employees/image21.png"));

    return data;
}

```



```

`javascript
function getLayoutInfo(node, tree) {
  if (!tree.hasSubTree) {
    tree.orientation = 'Vertical';
    tree.type = 'Right';
  }
}

function getNodeDefaults(obj, diagram) {
  obj.height = 100;
  obj.width = 250;
  obj.style = { fill: 'transparent', strokeWidth: 2 };
  obj.shape = {
    type: 'HTML',
    content:
      <div><div style=" width: 250px;background-color: #6BA5D7;height:100px; border: 2px solid
      darkblue; "></div><div style=" margin-left: 125px; margin-top: -74px; font-size: 15px;"> +
      (obj.data).Name +
      </div><div style=" margin-left: 125px; margin-top: 20px; font-size: 15px;"> +
      (obj.data).Designation +
      </div></div>
  };
}

function getConnectorDefaults(connector, diagram) {
  connector.targetDecorator.shape = 'None';
  connector.type = 'Orthogonal';
  connector.style.strokeColor = 'gray';
  return connector;
}

```

```
}
`
```

Dialog

Getting Started with ASP.NET MVC Dialog Control

This section briefly explains about how to include [ASP.NET MVC Dialog](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
`
```

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

```
`
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
```

```

...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>

```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```

<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>

```

Add ASP.NET MVC Dialog control

Now, add the Syncfusion ASP.NET MVC Dialog control in `~/Views/Home/Index.cshtml` page.


CSHTML

```

<div id="container" style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Dialog").Content("This is a Dialog
with content").Target("#container").Width("250px").Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0]
            dialog.show();
        }
    }
</script>

```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Dialog control will be rendered in the default web browser.



Note: In the dialog control, max-height is calculated based on the dialog target element height. If the target property is not configured, the document.body is considered as a target. Therefore, to show a dialog in proper height, you need to add min-height to the target element.

Modal Dialog

A [modal](#) shows an overlay behind the Dialog. So, the user should interact the Dialog compulsory before interacting with the remaining content in an application.

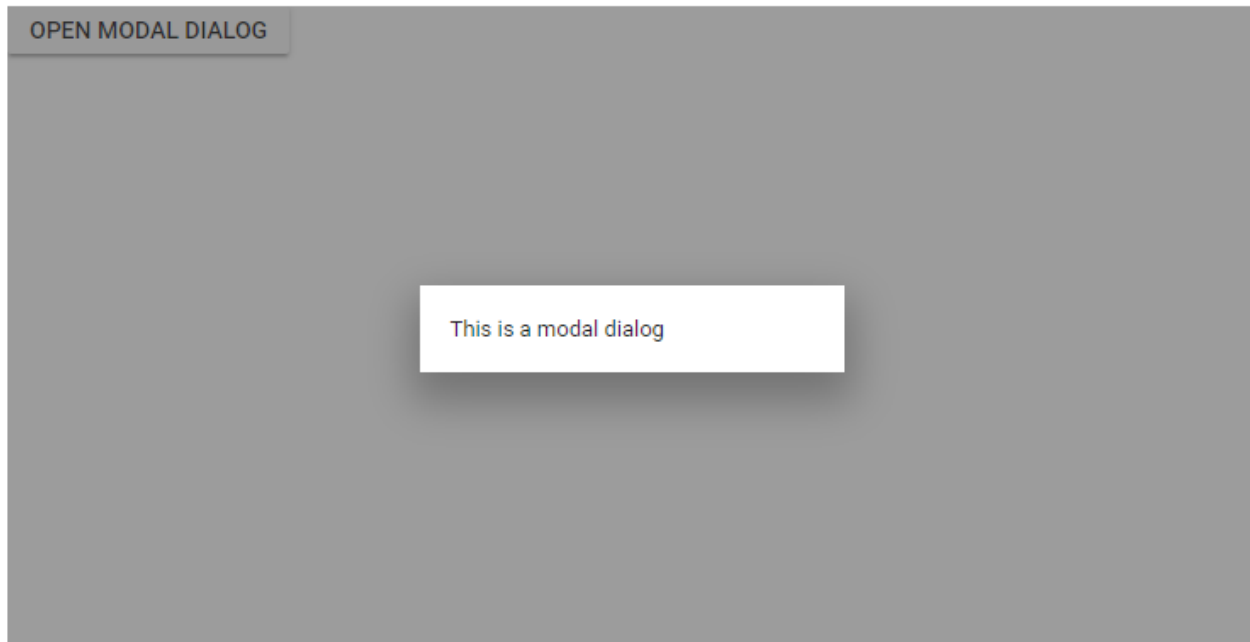
While the user clicks the overlay, the action can be handled through the [OverlayClick](#) event. In the below sample, the Dialog close action is performed while clicking on the overlay.

Note: When the modal dialog is opened, the Dialog's target scrolling will be disabled. The scrolling will be enabled again once close the Dialog.

CSHTML

```
<div id='container' style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Modal Dialog").Render()

    @Html.EJS().Dialog("dialog").IsModal(true).OverlayClick("onOverlayClick").Co
    ntent("This is a modal dialog").Width("300px").Target("#container").Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            dialog.show();
        }
    }
    function onOverlayClick() {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.hide();
    }
</script>
```

Note: In the dialog control, If the dialog is rendered based on the body, then the dialog get the height is based on its body element height. If the height of the dialog is larger than the body height, then the dialog's height will not be set. For this scenario, we can set the CSS style for the html and body to get the dialog height.

```
`css
```

```
html, body {  
height: 100%;  
}
```

```
,
```

Enable header

The Dialog header can be enabled by adding the header content as text or HTML content through the [Header](#) property.

CSHTML

```
<div id='container' style="height:400px;">  
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()  
  
    @Html.EJS().Dialog("dialog").Header("Dialog").ShowCloseIcon(true).Content("This is a dialog with header").Target("#container").Width("250px").Render()  
</div>  
<script>  
    window.onload = function () {  
        document.getElementById('targetButton').onclick = function () {  
            var dialog = document.getElementById("dialog").ej2_instances[0];  
            dialog.show();  
        }  
    }  
</script>
```

Enable footer

The Dialog provides built-in support to render the **buttons** on the footer (for ex: **OK** or **Cancel** buttons). Each Dialog button allows the user to perform any action while clicking on it.

The primary button will be focused automatically on open the Dialog, and add the [Click](#) event to handle the actions.

Note: When the Dialog initialize with more than one primary buttons, the first primary button gets focus on open the Dialog.

The below sample render with button and its click event.

CSHTML

```
<div id="container" style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Dialog").Content("This is a Dialog
with button and primary
button").Target("#container").Width("250px").Buttons(btn=> {
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
}).Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            dialog.show();
        }
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>
```

HOMEONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { isPrimary = true,
cssClass = "e-flat", content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
cssClass = "e-flat" };
        return View();
    }
}
```

Draggable

The Dialog supports to [drag](#) within its target container by grabbing the Dialog header, which allows the user to reposition the Dialog dynamically.

Note: The Dialog can be draggable only when the Dialog header is enabled. From **16.2.x** version, enabled draggable support for modal dialog also.

CSHTML

```
<div id='container' style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Dialog").Content("This is a Dialog
with drag
enabled").AllowDragging(true).Target("#container").Width("250px").AllowDragg
ing(true).Buttons(btn=> {
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
}).Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            dialog.show();
        }
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>
```

HOMECONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { cssClass = "e-flat",
content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
cssClass = "e-flat" };
        return View();
    }
}
```

Positioning

The Dialog position can be set through the [Position](#) property by providing X and Y coordinates. The Dialog can be positioned inside the target container based on the given X and Y values.

For example `<code>position:{ X:'center', Y:'center' }</code>` the possible values.

- for X is: left, center, right (or) any offset value
- for Y is: top, center, bottom (or) any offset value

The below sample demonstrates the different Dialog positions.

CSHTML

```
<div id='container' style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Dialog
Positions").FooterTemplate("<span id='posvalue' style='float:left; padding-
left:10px;font-weight: bold;'>Position: {X: 'left', Y:
'top'}</span>").Target("#container").Width("350px").CloseOnEscape(false).Con
tentTemplate(@<table style='width: 320px;'>
    <tr> <td><input type='radio' name='xy'
onclick='changePosition(event)' value='left top' checked='true'>left
top</td> <td><input type='radio' name='xy' onclick='changePosition(event)'
value='center top'>center top</td> <td><input type='radio' name='xy'
onclick='changePosition(event)' value='right top'>right top</td> </tr>
    <tr> <td><input type='radio' name='xy'
onclick='changePosition(event)' value='left center'>left center</td>
<td><input type='radio' name='xy' onclick='changePosition(event)'
value='center center'>center center</td> <td><input type='radio' name='xy'
onclick='changePosition(event)' value='right center'>right center</td> </tr>
    <tr> <td><input type='radio' name='xy'
onclick='changePosition(event)' value='left bottom'>left bottom</td>
<td><input type='radio' name='xy' onclick='changePosition(event)'
value='center bottom'>center bottom</td> <td><input type='radio' name='xy'
onclick='changePosition(event)' value='right bottom'>right bottom</td> </tr>
    </table>).Render()
</div>
<script>
function changePosition(event) {
    var dialog = document.getElementById("dialog").ej2_instances[0];
    dialog.position = { X: event.currentTarget.value.split(" ")[0], Y:
event.currentTarget.value.split(" ")[1] };
    document.getElementById("posvalue").innerHTML = 'Position: {X: "' +
event.currentTarget.value.split(" ")[0] + '", Y: "' +
event.currentTarget.value.split(" ")[1] + '"}';
};
</script>
```

Note: [View Sample in GitHub.](#)

See also

- [Real time example using Dialog](#)
- [Load dialog content using AJAX](#)
- [How to position the dialog on center of the page on scrolling](#)
- [Prevent closing of modal dialog](#)
- [Close dialog while click on outside of dialog](#)
- [How to make a reusable alert and confirm dialog](#)

Template

In Dialog the template support is provided to the header and footer sections. So any text or HTML content can be appending in these sections.

Header

The Dialog header content can be provided through the [Header](#) property, and it will allow both text and any HTML content as a string. Also in header, close button is provided as built-in support, and this can be enabled through the [ShowCloseIcon](#) property.

Footer

The Dialog footer can be enabled by adding built-in [Buttons](#) or providing any HTML string through the [FooterTemplate](#).

Note: The [Buttons](#) and [FooterTemplate](#) properties can't be used at the same time.

Content

The Dialog content can be provided through the [content](#) property, and it accepts both text and HTML string as content.

The below example demonstrates the usage of header, footer and content template in the Dialog

CSHTML

```
<div id="container" class="control-section" style="height:300px;">
    @Html.EJS().Button("targetButton").Content("OPEN DIALOG").Render()
    @Html.EJS().Dialog("dialog").CssClass("custom-
template").Created("onLoad").Header(
    "<img class='img2'
src='https://ej2.syncfusion.com/demos/src/dialog/images/1.png' alt='header
image'>" +
    "<div title='Nancy' class='dlg-template e-icon-settings'> Nancy
</div>").FooterTemplate(
    "<input id='inVal' class='e-input' type='text' placeholder='Enter your
message here!'>" +
    "<button id='sendButton' class='e-control e-btn e-primary' data-
ripple='true'>Send</button>").ShowCloseIcon(true).CloseOnEscape(false).Width
("45%").Target(
    "#container").Height("85%").ContentTemplate(@<div class='dialogContent'>
    <span class='dialogText'>Greetings Nancy! When will you share me the
source files of the project?</span>
    </div>).Render()
</div>
<style>
    #dialog .e-icon-settings::before {
        padding: 3px;
        font-size: 15px;
    }
    #sendButton {
        top: 5px;
        position: relative;
        right: 7px;
    }
    .custom-template.e-dialog .e-dlg-header-content {
        border-bottom: none;
        padding: 11px;
    }
</style>
```

```

.e-dialog .e-footer-content {
    right: 7px;
}
.e-dialog .e-dlg-header-content {
    background-color: #007DD1;
    padding: 10px;
}
    .e-dialog .e-dlg-header-content .e-btn.e-dlg-closeicon-btn {
        top: 5px;
        left: -11px;
    }
.e-dialog .e-dlg-header {
    position: relative;
}
.e-dialog .e-dlg-content {
    padding: 0;
}
#sendbtn {
    margin-right: -17px;
    margin-top: -2%;
}
.e-open-icon::before {
    content: '\e782';
}
.e-dialog .e-dlg-header > .img2 {
    height: 36px;
    width: 36px;
    border-radius: 50%;
    vertical-align: middle;
    display: inline-block;
}
.e-dialog .e-dlg-header .dlg-template {
    display: inline-block;
    padding: 0 10px;
    vertical-align: middle;
    height: 30px;
    line-height: 30px;
    color: #fff;
}
.e-dialog .e-dlg-header-content .e-btn .e-btn-icon {
    margin-top: -3px;
}
.custom-template .e-input {
    width: 75%;
    float: left;
    height: 18px;
}
.dialogContent .dialogText {
    font-size: 13px;
    padding: 5%;
    word-wrap: break-word;
    border-radius: 6px;
    text-align: justify;
    font-style: initial;
    display: block;
}
.dialogContent .dialogText {

```

```

        background-color: #f5f5f5;
    }
    .custom-template.e-dialog .e-footer-content {
        border-top: 0.5px solid #949494;
        padding-left: 31px;
    }
    .dialogContent {
        display: block;
        font-size: 15px;
        word-wrap: break-word;
        text-align: center;
        font-style: italic;
        border-radius: 6px;
        padding: 3%;
        position: relative;
        top: 25px;
    }
    .e-dialog .dialogContent {
        top: 20px;
    }
    .control-wrapper .e-control.e-dialog {
        width: 30%;
    }
    .e-dialog .e-dlg-header-content .e-icon-dlg-close {
        color: #fff;
    }
    .e-dialog .e-btn.e-dlg-closeicon-btn:hover span {
        color: #8ECBFF;
    }

    .e-dialog .e-dlg-header-content .e-btn.e-dlg-closeicon-btn:hover {
        background-color: rgba(255,255,255, 0.10);
    }
    .e-dialog .e-dlg-header-content .e-btn.e-dlg-closeicon-btn:focus {
        background-color: rgba(255,255,255, 0.10);
    }

    .e-dialog .e-dlg-header-content .e-btn.e-dlg-closeicon-btn:active .e-
icon-dlg-close {
        color: #fff;
    }
    .e-dialog .e-dlg-header-content .e-btn.e-dlg-closeicon-btn:focus .e-
icon-dlg-close {
        color: #fff;
    }
    .e-dialog .e-dlg-header-content .e-btn.e-dlg-closeicon-btn:hover .e-
icon-dlg-close {
        color: #fff;
    }
}
</style>
<script>
    var validText;
    function onLoad() {
        document.getElementById('targetButton').onclick = function () {
            var dialogObj =
document.getElementById('dialog').ej2_instances[0];
            dialogObj.show();

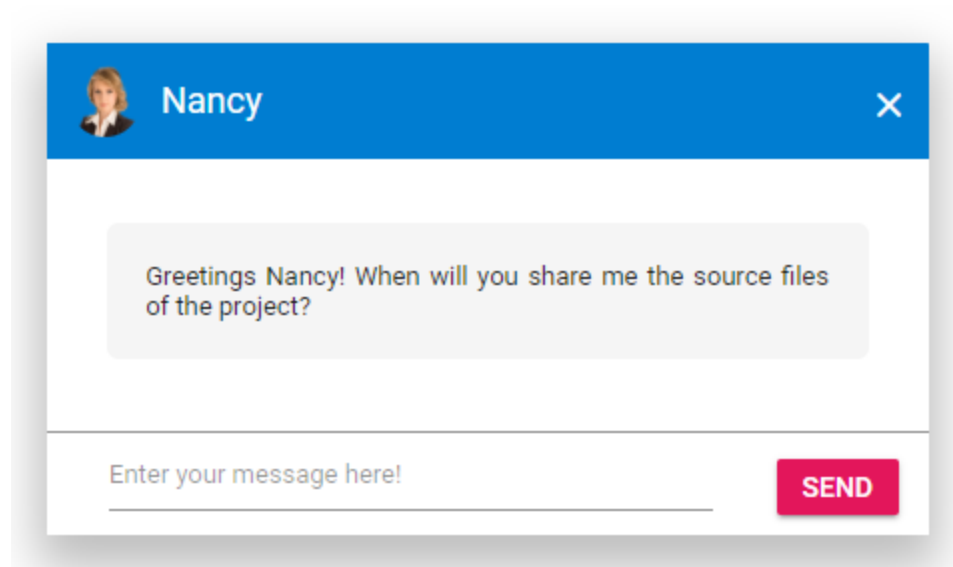
```

```
};
(document.getElementById('inVal')).onkeydown = function (e) {
    if (e.keyCode === 13 && validText !== "") {
        updateTextValue();
    }
};
(document.getElementById('sendButton')).onkeydown = function (e) {
    if (e.keyCode === 13 && validText !== "") {
        updateTextValue();
    }
};
document.getElementById('sendButton').onclick = function (e) {
    updateTextValue();
}
}
function updateTextValue() {
    var enteredVal = document.getElementById('inVal');
    var dialogTextElement =
document.getElementsByClassName('dialogText')[0];
    var dialogTextWrap =
document.getElementsByClassName('dialogContent')[0];
    if (enteredVal.value !== '') {
        dialogTextElement.innerHTML = enteredVal.value;
        enteredVal.value = '';
    }
}
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Output be like the below.



See Also

- [How to add an icon to dialog buttons](#)
- [How to customize the dialog appearance](#)

Animation

The Dialog can be animated during the open and close actions. Also, user can customize animation's [delay](#), [duration](#) and [effect](#).

<!-- markdownlint-disable MD033 -->

delay	The Dialog animation will start with the mentioned delay
duration	Specifies the animation duration to complete with one animation cycle
effect	<p>Specifies the animation effects of Dialog open and close actions effect.</p> <p>List of supported animation effects: 'Fade' 'FadeZoom' 'FlipLeftDown' 'FlipLeftUp' 'FlipRightDown' 'FlipRightUp' 'FlipXDown' 'FlipXUp' 'FlipYLeft' 'FlipYRight' 'SlideBottom' 'SlideLeft' 'SlideRight' 'SlideTop' 'Zoom' 'None'</p> <p>If the user sets 'Fade' effect, then the Dialog will open with 'FadeIn' effect and close with 'FadeOut' effect</p>

In the below sample, **Zoom** effect is enabled. So, The Dialog will open with **ZoomIn** and close with **ZoomOut** effects.

CSHTML

```
<div id='container' style="height:400px;">  
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
```

```
@Html.EJS().Dialog("dialog").Header("Dialog").AnimationSettings(e=>e.Effect(
Syncfusion.EJ2.Popups.DialogEffect.Zoom).Duration(400).Delay(0)).Content("Di
alog enabled with Zoom effect").Width("250px").Buttons(btn=> {
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
}).Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            dialog.show();
        }
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { cssClass = "e-flat",
content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
cssClass = "e-flat" };
        return View();
    }
}
```

Resizing

The Dialog supports resizing feature. To resize the dialog, we have to select and resize it by using its handle (grip) or hovering on any of the edges or borders of the dialog within the sample container.

The resizable dialog can be created by setting the `EnableResize` property to true, which is used to change the size of a dialog dynamically and view its content with expanded mode. The `ResizeHandles` property can also be configured for all the which directions in which the dialog should be resized. When you configure the target property along with the `EnableResize` property, the dialog can be resized within its specified target container.

CSHTML

```
<div id='container' style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
```

```

    @Html.EJS().Dialog("dialog").ResizeHandles(new List <string> () { "All"
    } ).Header("Dialog").EnableResize(true).Content("This is a Dialog with
    resize
    enabled").AllowDragging(true).Target("#container").Width("250px").Buttons(bt
    n=> {
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
    }).Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            dialog.show();
        }
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>

```

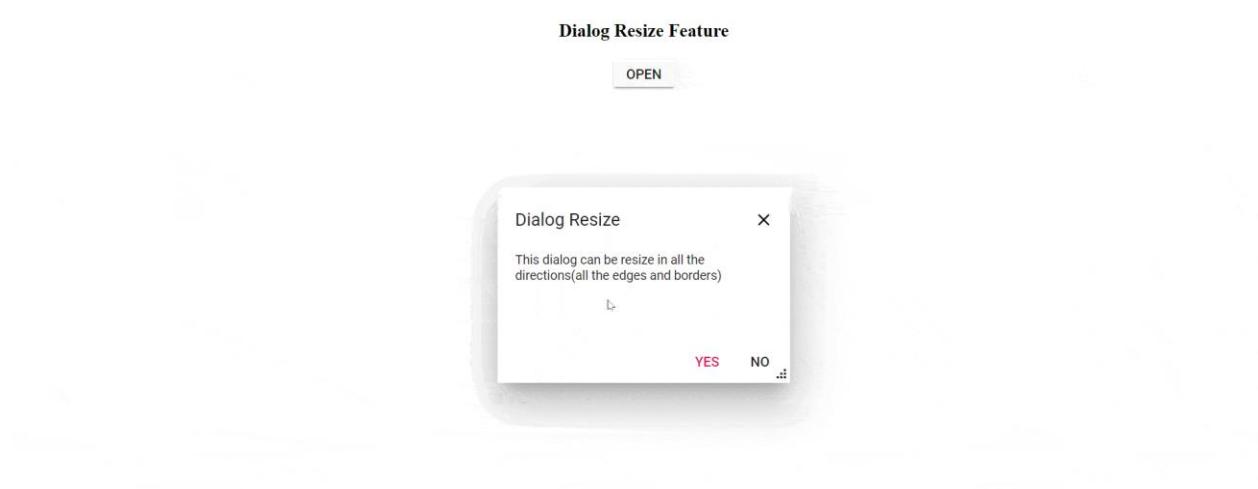
CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { cssClass = "e-flat",
        content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
        cssClass = "e-flat" };
        return View();
    }
}

```

Output be like the below.



CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the dialog header

Use the following CSS to customize the dialog header properties.

```
`CSS
.e-dialog .e-dlg-header {
color: green;
font-size: 20px;
font-weight: normal;
}
```

,

Customizing the dialog content

Use the following CSS to customize the dialog content properties.

```
`CSS
.e-dialog .e-dlg-content {
color: red;
font-size: 10px;
font-weight: normal;
line-height: normal;
}
```

,

Customizing modal dialog overlay

Use the following CSS to customize the modal dialog overlay.

```
`CSS
```

```
.e-dlg-overlay {  
background-color: slategray;  
opacity: 0.6;  
}  
`
```

Customizing the dialog resize icon

Use the following CSS to customize the dialog resize icon.

```
`CSS  
  
/ To change the icon content /  
.e-dialog .e-south-east::before, .e-dialog .e-south-west::before {  
content: '\f047';  
}  
  
/ To set the icon pack /  
.e-dialog .e-resize-handle {  
font: normal normal normal 14px/1 FontAwesome;  
}  
`
```

The above CSS demonstration uses the font awesome icon.

Customizing the dialog close button

Use the following CSS to customize the dialog close button.

```
`CSS  
  
/ To specify font size and color /  
.e-dialog .e-btn .e-btn-icon.e-icon-dlg-close {  
font-size: 12px;  
color: red;  
}  
`
```

Customizing the dialog footer button

Use the following CSS to customize the dialog footer button.

```
`CSS  
  
/ To specify font color, background color and border color /  
.e-btn.e-flat.e-primary, .e-css.e-btn.e-flat.e-primary {  
background-color: transparent;  
border-color: transparent;  
}
```

```
color: blue;
```

```
}
```

```
,
```

Localization

Localization library allows to localize the default text content of Dialog. In Dialog, The close button's tooltip text alone will be localize based on culture.

| Locale key | en-US (default) |

|-----|-----|

| close | close |

Loading translations

To load translation object in an application use `load` function of `L10n` class.

In the below sample, `French` culture is set to Dialog and change the close button's tooltip text.

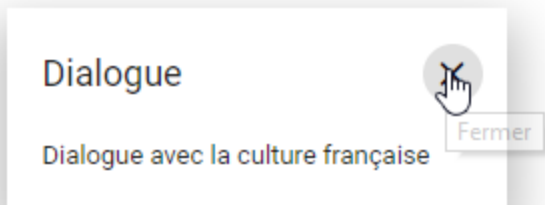
CSHTML

```
<div id="container" style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Ouvrir le
    dialogue").Render()
    @Html.EJS().Dialog("dialog").Header("Dialogue").Locale("fr-
    BE").Content("Dialogue avec la culture
    française").Target("#container").Width("250px").ShowCloseIcon(true).Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            dialog.show();
        }
        ej.base.L10n.load({
            'fr-BE': {
                'dialog': { 'close': "Fermer" }
            }
        });
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Output be like the below.



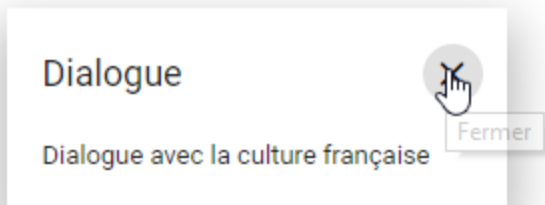
Accessibility

The Dialog component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dialog component is outlined below.

| Accessibility Criteria | Compatibility |

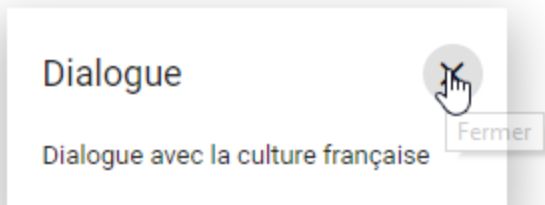
| -- | -- |

| [WCAG 2.2 Support](#) |  alt="Yes"> |

| [Section 508 Support](#) |  alt="Yes"> |

| [Screen Reader Support](#) |  alt="Yes"> |

| [Right-To-Left Support](#) |  alt="Yes"> |

| [Color Contrast](#) |  alt="Yes"> |

| [Mobile Device Support](#) |  alt="Yes"> |

| [Keyboard Navigation Support](#) |  alt="Yes"> |

| [Accessibility Checker Validation](#) |  alt="Yes"> |

| [Axe-core Accessibility Validation](#) |  alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Dialog characterized with complete ARIA Accessibility support which helps to accessible by on-screen readers and other assistive technology devices. This component designed with the reference of the guidelines document given in [WAI ARAI Accessibility Practices](#).

The Dialog control uses the **Dialog** role and following ARIA properties to its element based on its state.

| Property | Functionalities |

| --- | --- |

| aria-describedby | It indicates the Dialog content description is notified to the user through assistive technologies. |

| aria-labelledby | The Dialog title is notified to the user through assistive technologies. |

| aria-modal | For modal dialog it's value is true and non-modal dialog its value is false |

| aria-grabbed | Enable the draggable Dialog and starts dragging it is value is true and stopping the drag its value is false |

Keyboard interaction

Keyboard interaction of Dialog control has designed based on [WAI-ARIA Practices](#) described for Dialog. User can use the following shortcut keys to interact with the Dialog.

<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
Esc	Closes the Dialog. This functionality can be controlled by using closeOnEscape
Enter	When the Dialog button or any input (except text area) is in focus state, when pressing the Enter key, the click event associated with the primary button or button will trigger. Enter key is not working when the Dialog content contains any text area with initial focus
Ctrl + Enter	When the Dialog content contains text area and it is in focus state, and press the Ctrl + Enter key to call the click event function associated with primary button
Tab	Focus will be changed within the Dialog elements
Shift + Tab	The Focus will be changed previous focusable element within the Dialog elements. When focusing a first focusable element in the Dialog, then press the shift + tab key, it will change the focus to last focusable element

CSHTML


```

<div id='container' style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()

    @Html.EJS().Dialog("dialog").Header("Feedback").ShowCloseIcon(true).Target("#container").Width("400px").Buttons(btn=> {
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons).Add();
    }).Height("300px").ContentTemplate(@<form>
        <div class='form-group'>
            <label for='email'>Email:</label><input type='email' class='form-control' id='email'>
        </div><div class='form-group'>
            <div><div class='form-group'>
                <label for='comment'>Comments:</label><textarea style='resize:none;' class='form-control' rows='5' id='comment'></textarea>
            </div>
        </div></form>).Render()
    </div>
    <script>
        window.onload = function () {
            document.getElementById('targetButton').onclick = function () {
                var dialog = document.getElementById("dialog").ej2_instances[0];
                dialog.show();
            }
        }
        function dlgButtonClick() {
            var dialogObj = document.getElementById('dialog').ej2_instances[0];
            dialogObj.hide();
        }
    </script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons = new ButtonModel() { cssClass = "e-flat",
content = "Submit" };
        return View();
    }
}

```

See Also

- [Show dialog with full-screen](#)

Ensuring accessibility

The Dialog component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dialog component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dialog component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

How To

Create nested Dialog

A Dialog can be nested within another Dialog. The below sample contains parent and child Dialog (inner Dialog).

Step 1:

Create two div elements with id `#dialog` and `#innerDialog`.

Step 2:

Initialize the Dialog as mentioned in the below sample.

Step 3:

Set the inner Dialog target as `#dialog`.

CSHTML

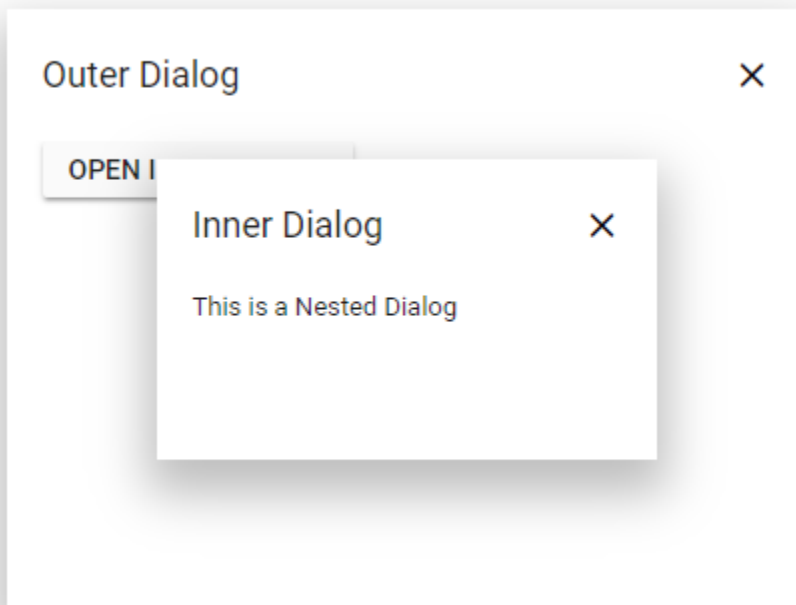
```
@using Syncfusion.EJ2.Popups
<div id='container' style='height:400px;'>
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Outer
Dialog").ShowCloseIcon(true).Target("#container").Height("300px").CloseOnEsc
ape(false).Width("400px").AnimationSettings(e =>
e.Effect(DialogEffect.None)).Created("onCreated").ContentTemplate(@<button
class="e-control e-btn" id="innerButton" role="button">Open
InnerDialog</button>).Render()
    @Html.EJS().Dialog("innerDialog").ShowCloseIcon(true).Header("Inner
Dialog").CloseOnEscape(false).Content("This is a Nested
Dialog").Target("#dialog").Height("150px").AnimationSettings(e =>
e.Effect(DialogEffect.None)).Width("250px").Render()
</div>
<script>
    window.onload = function () {
        document.getElementById('targetButton').onclick = function () {
            var dialog = document.getElementById("dialog").ej2_instances[0];
            var innerDialog =
document.getElementById("innerDialog").ej2_instances[0];
            dialog.show();
            innerDialog.show();
        }
    }
    function onCreated() {
        document.getElementById("innerButton").addEventListener("click",
function () {
            var innerDialog =
document.getElementById("innerDialog").ej2_instances[0];
            innerDialog.show();
        })
    }
```

```
}  
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller  
{  
    public ActionResult Index()  
    {  
        return View();  
    }  
}
```

Output be like the below.



Position the Dialog in center of the page on scrolling

By default, when scroll the page/container Dialog also scrolled along with the page/container. When a user expects to display the Dialog in the same position without scrolling achieving this in sample level as like below. Here added 'e-fixed' class to Dialog element and prevent the scrolling.

CSHTML

```
<div id='container' style="height:270px;overflow:auto">  
    <div>  
        <b>JavaScript:</b><br />  
        JavaScript is a high-level, dynamic, untyped, and interpreted  
programming language. It has been standardized in the ECMAScript  
        language specification. Alongside HTML and CSS, it is one of the  
three essential technologies of World Wide Web  
        content production; the majority of websites employ it and it is  
supported by all modern Web browsers without
```

plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object - oriented , imperative, and functional programming styles.

```
<br /><br /><br />
```

```
<b>MVC:</b><br />
```

Model-view-controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it. The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants. The controller mediates input, converting it to commands for the model or view. The central ideas behind MVC are code reusability and in addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).

A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.

A view requests from the model the information that it needs to generate an output representation to the user.

```
</div>
```

```
@Html.EJS().Dialog("dialog").Header("Dialog").Created("onCreated").CloseOnEscape(false).Target("#container").Width("250px").ContentTemplate(@<button class='e-control e-btn' id='targetButton' role='button'>Prevent Dialog Scroll</button>).Render()
```

```
</div>
```

```
<script>
```

```
function onCreated() {
    document.getElementById('targetButton').onclick = function () {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.cssClass = 'e-fixed';
    }
}
```

```
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Load Dialog content using AJAX

You can load dialog's content dynamically from external source like external file using AJAX library. The AJAX library can make the request and load dialog's content using its `success` event. Refer the following link to learn about how to load dialog content using AJAX.

AJAX Content

Render a Dialog using utility functions

The Dialog control provides built-in utility functions to render the alert and confirm dialogs with the minimal code. The following options are used as an argument on calling the utility functions:

Options	Description
title	Specifies the title of dialog like the header property.
content	Specifies the value that can be displayed in dialog's content area like the content property.
isModal	Specifies the Boolean value whether the dialog can be displayed as modal or non-modal. For more details, refer to the isModal property.
position	Specifies the value where the alert or confirm dialog is positioned within the document. For more details, refer to the position property { X: 'center', Y: 'center' }
okButton	Configures the OK button that contains button properties with the click events. okButton:{ icon:'prefix icon to the button', cssClass:'custom class to the button', click: 'action for OK button click', text: 'Yes' // <-- Default value is 'OK' }
cancelButton	Configures the Cancel button that contains button properties with the click events. cancelButton:{ icon:'prefix icon to the button', cssClass:'custom class to the button', click: 'action for 'Cancel' button click', text: 'No' // <-- Default value is 'Cancel' }
isDraggable	Specifies the value whether the alert or confirm dialog can be dragged by the user.
showCloseIcon	When set to true, the close icon is shown in the Dialog control.
closeOnEscape	When set to true, you can close the dialog by pressing ESC key.
cssClass	Specifies the CSS class name that can be appended to the dialog.
zIndex	Specifies the order of the dialog, that is displayed in front or behind of another component.
open	Event which is triggered after the dialog is opened.
close	Event which is triggered after the dialog is closed.
animationSettings	Specifies the animation settings of the dialog component.

Alert dialog

An alert dialog box is used to display warning like messages to the users. Use the following code to render a simple alert dialog in an application.

CSHTML

```
<div style="height:400px;">
    @Html.EJS().Button("dialogButton").Content("Open Alert Dialog").Render()
</div>
<script>
```

```
// To Render dialog utility on button click
document.getElementById('dialogButton').onclick = function () {
    // Initialize and render alert dialog
    ej.popups.DialogUtility.alert('This is an Alert Dialog!');
};
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Render an alert dialog with options

CSHTML

```
<div style="height:400px;">
    @Html.EJS().Button("dialogButton").Content("Open Alert Dialog").Render()
</div>
<script>
    // To Render dialog utility on button click
    document.getElementById('dialogButton').onclick = function () {
        // Initialize and render alert dialog with options
        ej.popups.DialogUtility.alert({
            title: 'Alert Dialog',
            content: "This is an Alert Dialog!",
            okButton: { text: 'OK', click: okClick },
            showCloseIcon: true,
            closeOnEscape: true,
            animationSettings: { effect: 'Zoom' }
        });
    };
    function okClick() {
        alert('You clicked OK button');
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Confirm dialog

A confirm dialog displays a specified message along with 'OK' and 'Cancel' button.

CSHTML

```
<div style="height:400px;">
    @Html.EJS().Button("dialogButton").Content("Open Confirm
Dialog").Render()
</div>
<script>
    // To Render dialog utility on button click
    document.getElementById('dialogButton').onclick = function () {
        // Initialize and render confirm dialog
        ej.popups.DialogUtility.confirm('This is a Confirmation Dialog!');
    };
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Render a confirmation dialog with options

CSHTML

```
<div style="height:400px;">
    @Html.EJS().Button("dialogButton").Content("Open Confirm
Dialog").Render()
</div>
<script>
    // To Render dialog utility on button click
    document.getElementById('dialogButton').onclick = function () {
    // Initialize and render confirm dialog with custom options
        ej.popups.DialogUtility.confirm({
            title: ' Confirmation Dialog',
            content: "This is a Confirmation Dialog!",
            okButton: { text: 'OK', click: okClick },
            cancelButton: { text: 'Cancel', click: cancelClick },
            showCloseIcon: true,
            closeOnEscape: true,
            animationSettings: { effect: 'Zoom' }
        });
        function okClick(){
            alert('You clicked OK button');
        }
        function cancelClick() {
            alert('You clicked Cancel button');
        }
    };
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Close utility dialog

When rendering an Alert and Confirmation dialog through utility methods, You can close the dialog using the following ways.

- By pressing the escape key if the [closeOnEscape](#) property is enabled.
- By clicking the close button if the [showCloseIcon](#) property is enabled.

You can also manually close the Dialogs by creating an instance to the dialog and call the "hide" method.

Below sample demonstrates the different ways of hiding the utility dialog.

CSHTML

```
<div style="height:400px;">
    @Html.EJS().Button("dialogButton").Content("Open Confirm
Dialog").Render()
</div>
<script>
    var DialogObj;
    // To Render dialog utility on button click
    document.getElementById('dialogButton').onclick = function () {
    // Initialize and render confirm dialog with custom options
    DialogObj = ej.popups.DialogUtility.confirm({
        title: ' Confirmation Dialog',
        content: "This is a Confirmation Dialog!",
        okButton: { text: 'OK', click: okClick },
        cancelButton: { text: 'Cancel', click: cancelClick },
        showCloseIcon: true,
        closeOnEscape: true,
        animationSettings: { effect: 'Zoom' }
    });
    function okClick() {
        alert('You clicked OK button');
    }
    function cancelClick() {
        //Hide the dialog
        DialogObj.hide();
    }
    };
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
    }
```



```

        return View();
    }
}

```

Add minimize and maximize buttons to the Dialog header

ASP.NET MVC Dialog allows end users to either minimize or maximize the Dialog component. You can add minimize and maximize custom buttons near the close icon in the Dialog header using the `headerTemplate` property and handle the actions in the button click events, as shown in the following sample.

CSHTML

```

@using Syncfusion.EJ2.Popups
<div id="target" class="col-lg-12 control-section">
    @Html.EJS().Button("normalbtn").Content("Open").Render()

    @Html.EJS().Dialog("dialog").AnimationSettings(a=>a.Effect(DialogEffect.Zoom))
    .Header("<div> <span class='title'>Dialog</span>
        <span class='e-icons sf-icon-Maximize' id='max-btn'
        title='Maximize'></span>
        <span class='e-icons sf-icon-Minimize' id='min-btn'
        title='Minimize'></span></div>").Content("This is a dialog with minimize and
        maximize
        buttons").ShowCloseIcon(true).CloseOnEscape(true).Width("500px").Target("#ta
        rget").Buttons(btn=> {

        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DefaultButtons1).Add();

        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DefaultButtons2).Add();
    }).Render()
</div>
<script>
    var isFullScreen;
    var dialogOldPositions;
    document.getElementById('max-btn').onclick = function () {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        var maximizeIcon;
        if (dialogObj.element.classList.contains('dialog-minimized')) {
            dialogObj.element.querySelector('#min-btn').classList.add('sf-icon-
            Minimize');
            dialogObj.element.querySelector('#min-btn').classList.remove('sf-
            icon-Restore');
            dialogObj.element.querySelector('#min-btn').setAttribute('title',
            'Minimize');
        }
        if (!dialogObj.element.classList.contains('dialog-maximized') &&
        !isFullScreen) {
            maximizeIcon = dialogObj.element.querySelector(".e-dlg-header-
            content .sf-icon-Maximize");
        } else {
            maximizeIcon = dialogObj.element.querySelector(".e-dlg-header-
            content .sf-icon-Restore");
        }
        if (!dialogObj.element.classList.contains('dialog-maximized')) {
            dialogObj.element.classList.add('dialog-maximized');
        }
    }
}

```

```

        dialogObj.show(true);
        maximizeIcon.classList.add('sf-icon-Restore');
        maximizeIcon.setAttribute('title', 'Restore');
        maximizeIcon.classList.remove('sf-icon-Maximize');
        dialogObj.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        isFullScreen = true;
    } else {
        dialogObj.element.classList.remove('dialog-maximized');
        dialogObj.show(false);
        maximizeIcon.classList.remove('sf-icon-Restore');
        maximizeIcon.classList.add('sf-icon-Maximize');
        maximizeIcon.setAttribute('title', 'Maximize');
        dialogObj.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        dialogObj.position = dialogOldPositions;
        dialogObj.dataBind();
        isFullScreen = false;
    }
}
document.getElementById('min-btn').onclick = function () {
    var dialogObj = document.getElementById('dialog').ej2_instances[0];
    var minimizeIcon = dialogObj.element.querySelector(".e-dlg-header-
content .sf-icon-Minimize");
    if (!dialogObj.element.classList.contains('e-dlg-fullscreen')) {
        if (!dialogObj.element.classList.contains('dialog-minimized')) {
            dialogOldPositions = { X: dialogObj.position.X, Y:
dialogObj.position.Y }
            dialogObj.element.classList.add('dialog-minimized');
            dialogObj.element.classList.remove('dialog-maximized');
            dialogObj.element.querySelector('.e-dlg-
content').classList.add('hide-content');
            dialogObj.position = { X: 'center', Y: 'bottom' };
            dialogObj.dataBind();
            minimizeIcon.classList.add('sf-icon-Restore');
            minimizeIcon.setAttribute('title', 'Restore');
        } else {
            dialogObj.element.classList.remove('dialog-minimized');
            dialogObj.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
            minimizeIcon.classList.add('sf-icon-Minimize');
            minimizeIcon.setAttribute('title', 'Minimize');
            minimizeIcon.classList.remove('sf-icon-Restore');
            dialogObj.position = dialogOldPositions;
            dialogObj.dataBind();
        }
    } else {
        dialogObj.show(false);
        dialogObj.element.classList.remove('dialog-maximized');
        dialogObj.element.classList.add('dialog-minimized');
        dialogObj.element.querySelector('.e-dlg-
content').classList.add('hide-content');
        minimizeIcon.classList.remove('sf-icon-Minimize');
        minimizeIcon.removeAttribute('title');
        dialogObj.position = { X: 'center', Y: 'bottom' };
        dialogObj.dataBind();
        isFullScreen = true;
    }
}

```

```

    }
}
document.getElementById('normalbtn').onclick = function () {
    var dialogObj = document.getElementById('dialog').ej2_instances[0];
    dialogObj.show();
};
function dlgButtonClick() {
    var dialogObj = document.getElementById('dialog').ej2_instances[0];
    dialogObj.hide();
}
</script>
<style>
    .control-section {
        padding-left: 10px;
    }
    #target {
        height: 100%;
        min-height: 350px;
    }
    .e-ok-icon::before {
        content: '\e7ff';
    }
    .e-close-icon::before {
        content: '\e825';
    }
    .control-section {
        height: 100%;
        min-height: 350px;
    }
    .dialog-minimized {
        top: 261px !important;
    }
    .control-section {
        width: 450px;
        height: 500px;
        border: 1px solid black;
    }
    .e-dialog .e-dlg-header {
        width: auto;
    }
    .e-dialog .e-dlg-header .e-icons.sf-icon-Maximize::before,
    .e-dialog .e-dlg-header .e-icons.sf-icon-Minimize::before,
    .e-dialog .e-dlg-header .e-icons.sf-icon-Restore::before {
        position: relative;
    }
    .e-dialog .e-dlg-header .e-icons.sf-icon-Minimize,
    .e-dialog .e-dlg-header .e-icons.sf-icon-Maximize,
    .e-dialog .e-dlg-header .e-icons.sf-icon-Restore {
        font-size: 14px;
        width: 30px;
        height: 30px;
        line-height: 30px;
        float: right;
        position: relative;
        text-align: center;
        cursor: pointer;
    }
}

```

```

.e-dialog .e-dlg-header .e-icons.sf-icon-Minimize:hover, .e-dialog .e-
dlg-header .e-icons.sf-icon-Maximize:hover,
.e-dialog .e-dlg-header .e-icons.sf-icon-Restore:hover {
    background-color: #e0e0e0;
    border-color: transparent;
    box-shadow: 0 0 0 transparent;
    border-radius: 50%;
}
.e-dialog .e-dlg-header .e-icons.sf-icon-Minimize,
.e-dialog .e-dlg-header .e-icons.sf-icon-Restore {
    padding-left: 5px;
    padding-right: 5px;
}

.e-dialog .e-dlg-header {
    position: relative;
    top: 1px;
}

.e-dialog .e-dlg-content.hide-content, .e-dialog .e-footer-content.hide-
content {
    display: none;
}

.e-dialog .e-dlg-header span.title {
    width: 60px;
    display: inline-block;
}
@font-face {
    font-family: 'Min-Max_FONT';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfUAAAEoAAAAVmNtYXDNdE+dkAAABlAAAAADxnbHlmQCk
X6AAAAwAAADkaGVhZBK7D5EAAADQAAANmhoZWEIVQQGAAAArAAAACRobXR4FAAAAAAAYAAAAA
UbG9jYQBaAJwAAAHQAAAADG1heHABEGAgAAABCAAAACBuYW1l8Rnd5AAAAsAAAAJhcG9zdDbKxec
AAAUkAAAATwABAAAEAAAAAFwEAAAAAAD+AAABAAAAAAAAAAAAAAAAABQABAAAAQAAK4KTXV8
PPPUACwQAAAAANfSZU4AAAAA19JlTgAAAAAD+AP4AAAACAACAAAAAAAAAAAAEAAAFABQAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAAAA4AKgBMAHIAAQAAAAADkwIyAAMAABMhNSFtAyb82gHOZAAAAAM
AAAAA/gD+AADAAcACwAAAREhESUVITUDIREhA5P82gMm/Np1A/D8EALK/aMCXcl1Zfx1A/AAAQA
AAADkwOSAAaABMJARcJATcJAScJAWwBTF6zRwFNAU1H/rMBTUf+s/6zA0v+tf61RwFL/rVHAUs
BS0f+tQFLAAADAAAAAP4A/gABQALABMAABMzIREhESURIXehNQCjESE1MxEh0mQB1P2jAyZ1/gh
kygMmyvzaAsr9owJdyf2jAfh1ZfzaygMmAAAAAASAN4AAQAAAAAAAAABAAAAQAAAAAAAAQAMAAE
AAQAAAAAAgAHAA0AAQAAAAAAAwAMABQAAQAAAAAABAAMACAAQAAAAAABQALACwAAQAAAAAABgA
MADcAAQAAAAAACgAsAEMAAQAAAAAACwASAG8AAwABBAkAAAACAIEAAwABBAkAAQAYAIMAAwABBAk
AAgAOAJsAAwABBAkAAwAYAKkAAwABBAkABAAYAMEAAwABBAkABQAWANkAAwABBAkABgAYAO8AAwA
BBAkACgBYAQcAAwABBAkACwAkAV8gTWluLU1heF9GT05UUmVndWxhcmlpbilNYXhfrk9OVELpbil
NYXhfrk9OVFZ1cnNpb24gMS4wTWluLU1heF9GT05UUmVndWxhcmlpbilNYXhfrk9OVELpbilNYXh
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIAABNAGkAbgAtAE0AYQB4AF8ARgB
PAE4AVABSAGUAZwB1AGwAYQByAE0AaQBuAC0ATQBhAHgAXwBGAE8ATgBUAE0AaQBuAC0ATQBhAHg
AXwBGAE8ATgBUAFYAZQByAHMAaQBVAG4AIAAxAC4AMABNAGkAbgAtAE0AYQB4AF8ARgBPAE4AVAB
GAG8AbgB0ACAAZwB1AG4AZQByAGEADAB1AGQAIAB1AHMAaQBuAGcAIABTAKAbgBjAGYAdQBzAGk
AbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwB
uAC4AYwBvAG0AAAAAGAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgA
ITWluaWlpemUITWF4aWlpemUFQ2xvc2UHUHVndG9yZQAAAA=) format('trueType');
    font-weight: normal;

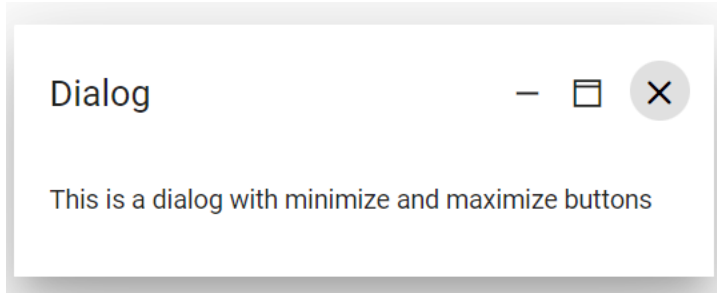
```

```
        font-style: normal;
    }
    [class^="sf-icon-"], [class*=" sf-icon-"] {
        font-family: 'Min-Max_FONT' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
    .sf-icon-Minimize:before {
        content: "\e700";
    }
    .sf-icon-Maximize:before {
        content: "\e701";
    }
    .sf-icon-Close:before {
        content: "\e702";
    }
    .sf-icon-Restore:before {
        content: "\e703";
    }
}
</style>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
        public string iconCss { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DefaultButtons1 = new ButtonModel() { content = "YES",
isPrimary = true, iconCss = "e-icons e-ok-icon" };
        ViewBag.DefaultButtons2 = new ButtonModel() { content = "NO",
iconCss = "e-icons e-close-icon" };
        return View();
    }
}
```

Output be like the below.



Render a Dialog without header

The dialog can be rendered without header by setting the [Header](#) property value as empty string or null. By default, dialog is rendered without header.

CSHTML

```
<div id="container" style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Content("This is a dialog without
header.").Target("#container").Width("250px").Buttons(btn=> {
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
}).Render()
</div>
<script>
    document.getElementById('targetButton').onclick = function () {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.show();
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { isPrimary = true,
cssClass = "e-flat", content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
cssClass = "e-flat" };
        return View();
    }
}
```

Show Dialog with fullscreen

You can show the dialog in fullscreen by passing `true` as argument to the dialog `show` method.

CSHTML

```
<div id="container">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()

    @Html.EJS().Dialog("dialog").Header("Dialog").Visible(false).Content("This
    is a Dialog with fullscreen
    display.").Target("#container").Width("250px").Buttons(btn=>
    {
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
        btn.Click("dlgButtonCancel").ButtonModel(ViewBag.DialogButtons2).Add();
    }).Render()
</div>
<script>
    document.getElementById('targetButton').onclick = function () {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.show(true);
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>
<style>
    #container {
        height: 100%;
        width: 100%;
        position: absolute;
        padding: 20px;
        margin: 0px;
        left: 0px;
        right: 0px;
        bottom: 0px;
        top: 0px;
    }
</style>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { isPrimary = true,
        cssClass = "e-flat", content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
        cssClass = "e-flat" };
    }
}
```

```
        return View();  
    }  
}
```

Display a Dialog with custom position

By default, the dialog is displayed in the center of the target container. The dialog position can be set using the [Position](#) property by providing custom X and Y coordinates. The dialog can be positioned inside the target based on the given X and Y values.

CSHTML

```
<div id="container" style="height:400px;">  
    @Html.EJS().Dialog("firstDialog").Header("Position-01").Content("The  
    dialog is positioned at {X: 160, Y: 14} co-ordinates.").Position(p =>  
    p.X(160).Y(12)).Target("#container").Width("360px").Height("120px").Render()  
    @Html.EJS().Dialog("secondDialog").Header("Position-02").Content("The  
    dialog is positioned at {X: 160, Y: 240} co-ordinates.").Position(p =>  
    p.X(160).Y(240)).Target("#container").Width("360px").Height("120px").Render(  
    )  
</div>
```

CONTROLLER.CS

```
public class HomeController : Controller  
{  
    public ActionResult Index()  
    {  
        return View();  
    }  
}
```

Output be like the below.

Position-01

The dialog is positioned at {X: 160, Y: 14} coordinates

Position-02

The dialog is positioned at {X: 160, Y: 240} coordinates

Prevent closing of modal Dialog

You can prevent closing of modal dialog by setting the [BeforeClose](#) event argument cancel value to true. In the following sample, the dialog is closed when you enter the username value with minimum 4 characters. Otherwise, it will not be closed.

CSHTML

```
<div id="container" style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Sign
In").BeforeClose("validation").ContentTemplate(@<div class='wrap'><div
id='input-container'>
    <div class='e-float-input'>
        <input id='textvalue' type='text' required /><span class='e-float-
line'></span>
        <label class='e-float-text'>Username </label>
    </div>
</div><div class='form-group'>
    <div class='e-float-input'>
        <input id='textvalue2' type='Password' required />
        <span class='e-float-line'></span>
        <label class='e-float-text'>Password</label>
    </div>
</div></div>).Target("#container").Width("300px").Buttons(btn=>{
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons).Add();
}).Render()
<script>
    // To get the all input fields and its container.
```

```

    let inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
    // Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
    for (let i = 0; i < inputElement.length; i++) {
        inputElement[i].addEventListener("focus", function () {
            let parentElement = this.parentNode;
            if (parentElement.classList.contains('e-input-in-wrap')) {
                parentElement.parentNode.classList.add('e-input-focus');
            } else {
                this.parentNode.classList.add('e-input-focus');
            }
        });
        inputElement[i].addEventListener("blur", function () {
            let parentElement = this.parentNode;
            if (parentElement.classList.contains('e-input-in-wrap')) {
                parentElement.parentNode.classList.remove('e-input-focus');
            } else {
                this.parentNode.classList.remove('e-input-focus');
            }
        });
    }
    // Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
    var inputIcon = document.querySelectorAll('.e-input-group-icon');
    for (let i = 0; i < inputIcon.length; i++) {
        inputIcon[i].addEventListener('mousedown', function () {
            this.classList.add('e-input-btn-ripple');
        });
        inputIcon[i].addEventListener('mouseup', function () {
            let element = this;
            setTimeout(function () {
                element.classList.remove('e-input-btn-ripple');
            }, 500);
        });
    }
    document.getElementById('targetButton').onclick = function () {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.show();
        document.getElementById("textvalue").value = "";
        document.getElementById("textvalue2").value = "";
    };
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
    function validation(args) {
        let text = document.getElementById('textvalue');
        let text1 = document.getElementById('textvalue2');
        if (text.value === "" && text1.value === "") {
            args.cancel = true;
            alert("Enter the username and password")
        } else if (text.value === "") {
            args.cancel = true;
            alert("Enter the username")
        } else if (text1.value === "") {
            args.cancel = true;

```

```

        alert("Enter the password")
    } else if (text.value.length < 4) {
        args.cancel = true;
        alert("Username must be minimum 4 characters")
    } else {
        args.cancel = false;
        document.getElementById("textvalue").value = "";
        document.getElementById("textvalue2").value = "";
    }
}
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons = new ButtonModel() { isPrimary = true,
content = "LOG IN" };
        return View();
    }
}

```

Prevent the focus on the first element

By default, the dialog focuses on the first elements of the content area which can be active and focusable. You can prevent this default focusing behavior using the [Open](#) event and by enabling the `preventFocus` argument.

Bind the open event and enable the `preventFocus` argument within an event like the below example.

CSHTML

```

<div id="container" style="height:400px;">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()
    @Html.EJS().Dialog("dialog").Header("Sign
In").Open("onOpen").ContentTemplate(@<div class='wrap'>
    <div class='form-group'><label for='email'>Email:</label>
    <input type='email' class='form-control' id='email'>
    </div>
    <div class='form-group'>
    <label for='comment'>Password:</label>
    <input type='password' class='form-control' id='password'>
    </div></div>).Target("#container").Width("300px").Buttons(btn=> {
btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();}).Ren
der()
</div>

```

```

<script>
    document.getElementById('targetButton').onclick = function () {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.show();
    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
    function onOpen() {
        args.preventDefault = true;
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { isPrimary = true,
content = "Ok" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel" };
        return View();
    }
}

```

Prevent opening of the dialog

You can prevent opening of the dialog by setting the [BeforeOpen](#) event argument cancel value to true. In the following sample, the success dialog is opened when you enter the username value with minimum 4 characters. Otherwise, it will not be opened.

CSHTML

```

<div id="container">
    <div class="login-form">
        <div class='wrap'>
            <div id="heading">Sign in</div>
            <div id="input-container">
                <div class="e-float-input e-input-group">
                    <input id="textvalue" type="text" required />
                    <span class="e-float-line"></span>
                    <label class="e-float-text">Username</label>
                </div>
                <div class="e-float-input e-input-group">
                    <input id="textvalue2" type="password" required />
                    <span class="e-float-line"></span>
                    <label class="e-float-text">Password</label>
                </div>
            </div>
        </div>
    </div>

```

```

        <div class="button-contain">
            <button class="e-control e-btn e-info" id="targetButton"
role="button" e-ripple="true">Log in</button>
        </div>
    </div>
</div>

@Html.EJS().Dialog("dialog").Header("Dialog").Created("onCreated").BeforeOpen("validation").Visible(false).Content("This is a Dialog with full screen display.").Target("#container").Width("250px").Buttons(btn=> {
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons).Add();
}).Render()
</div>
<script>
    // To get the all input fields and its container.
    var dialogObj;
    function onCreated() {
        dialogObj = this;
    }
    let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-float-input.e-input-group input');
    // Add 'e-input-focus' class to the input for achive ripple effect when focus on the input field.
    for (let i = 0; i < inputElement.length; i++) {
        inputElement[i].addEventListener("focus", function () {
            let parentElement = this.parentNode;
            if (parentElement.classList.contains('e-input-in-wrap')) {
                parentElement.parentNode.classList.add('e-input-focus');
            } else {
                this.parentNode.classList.add('e-input-focus');
            }
        });
        inputElement[i].addEventListener("blur", function () {
            let parentElement = this.parentNode;
            if (parentElement.classList.contains('e-input-in-wrap')) {
                parentElement.parentNode.classList.remove('e-input-focus');
            } else {
                this.parentNode.classList.remove('e-input-focus');
            }
        });
    }
    // Add 'e-input-btn-ripple' class to the icon element for achive ripple effect when click on the icon.
    var inputIcon = document.querySelectorAll('.e-input-group-icon');
    for (let i = 0; i < inputIcon.length; i++) {
        inputIcon[i].addEventListener('mousedown', function () {
            this.classList.add('e-input-btn-ripple');
        });
        inputIcon[i].addEventListener('mouseup', function () {
            let element = this;
            setTimeout(function () {
                element.classList.remove('e-input-btn-ripple');
            }, 500);
        });
    }
    document.getElementById('targetButton').onclick = function () {
        dialogObj.show();
    }

```

```

    };
    function dlgButtonClick() {
        dialogObj.hide();
    }
    function validation(args) {
        let text = document.getElementById('textvalue');
        let text1 = document.getElementById('textvalue2');
        if (text.value === "" && text1.value === "") {
            args.cancel = true;
            alert("Enter the username and password")
        } else if (text.value === "") {
            args.cancel = true;
            alert("Enter the username")
        } else if (text1.value === "") {
            args.cancel = true;
            alert("Enter the password")
        } else if (text.value.length < 4) {
            args.cancel = true;
            alert("Username must be minimum 4 characters")
        } else {
            args.cancel = false;
            document.getElementById("textvalue").value = "";
            document.getElementById("textvalue2").value = "";
        }
    }
}
</script>
<style>
    html,
    body,
    #container {
        height: 100%;
        overflow: hidden;
        width: 100%;
    }
    #dialog.e-dialog .e-dlg-header-content .e-dlg-header {
        text-align: center;
        width: 100%;
        color: #333;
        font-weight: bold;
        font-size: 20px;
    }
    #input-container .e-float-input { /* csslint allow: adjoining-classes */
        margin: 17px 0;
    }
    .text-center {
        text-align: center;
    }
    .e-footer-content {
        padding: 20px 0 0;
        width: 100%;
    }
    .e-dialog .e-footer-content .e-btn {
        width: 100%;
        height: 36px;
    }
    #heading {
        color: #333;

```

```

        font-weight: bold;
        margin: 0 0 15px;
        text-align: center;
        font-size: 20px;
    }
    .wrap {
        box-sizing: border-box;
        margin: 0 auto;
        width: 260px;
    }
    #dialog.e-dialog .e-footer-content {
        padding: 0 18px 18px;
    }
    #dialog.e-dialog .e-footer-content .e-btn {
        margin-left: 0;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons = new ButtonModel() { isPrimary = true,
content = "Dismiss" };
        return View();
    }
}

```

Read all values of Dialog on button click

You can read the dialog element values by binding the action handler to the footer buttons. The [Buttons](#) property provides the options to bind events to the action buttons. For detailed information about buttons, refer to the [footer](#) section. In the below sample, value of input elements within the dialog has been checked in the footer button click event and send the values as the content of confirmation dialog.

CSHTML

```

@using Syncfusion.EJ2.Popups
<div id="target" class="col-lg-12 control-section">
    @Html.EJS().Button("normalbtn").Content("Open").Render()

    @Html.EJS().Dialog("dialog").AnimationSettings(a=>a.Effect(DialogEffect.Zoom))
    .Header("User Details").ShowCloseIcon(true).Width("400px").Target("#target").Buttons(btn=>
    {
        btn.Click("submitButtonClick").ButtonModel(ViewBag.DialogButtons).Add();
    }).ContentTemplate(@<form>
        <div class="form-group"><label for="name">Name:</label><input
type="name" value="" class="form-control" id="name"></div>

```

```

        <div class="form-group"><label for="email">Email Id:</label><input
type="email" value="user@syncfusion.com" class="form-control"
id="email"></div>
        <div class="form-group"><label for="contact">Contact
Number:</label><input type="contact" class="form-control"
id="contact"></div>
        <div class="form-group"><label
for="address">Address:</label><textarea class="form-control" rows="5"
id="address"></textarea></div>
    </form>).Render()
    @Html.EJS().Dialog("model_dialog").AnimationSettings(a =>
a.Effect(DialogEffect.Zoom)).Header("User
Details").Visible(false).ShowCloseIcon(true).IsModal(true).Target("#target")
.Buttons(btn=> {
    btn.Click("dlgButtonClick").ButtonModel(ViewBag.SubmitButtons1).Add();
    btn.Click("returnClick").ButtonModel(ViewBag.SubmitButtons2).Add();
}).Render()
</div>
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<script>
    document.getElementById("normalbtn").onclick = function () {
        var dialogObj = document.getElementById("dialog").ej2_instances[0];
        dialogObj.show();
    };
    function dlgButtonClick() {
        var dialogObj =
document.getElementById("model_dialog").ej2_instances[0];
        dialogObj.hide();
    }
    function returnClick() {
        var dialogObj =
document.getElementById("model_dialog").ej2_instances[0];
        dialogObj.hide();
        var dialogObj1 = document.getElementById("dialog").ej2_instances[0];
        dialogObj1.show();
    }
    function submitButtonClick() {
        var dialogObj = document.getElementById("dialog").ej2_instances[0];
        dialogObj.hide();
        var dialogObj1 =
document.getElementById("model_dialog").ej2_instances[0];
        dialogObj1.content = getDynamicContent();
        dialogObj1.show();
    }
    function getDynamicContent() {
        var input =
document.getElementById('dialog').querySelector('#name');
        var email =
document.getElementById('dialog').querySelector('#email');
        var contact =
document.getElementById('dialog').querySelector('#contact');
        var address =
document.getElementById('dialog').querySelector('#address');
        var template = "<div class='row'><div class='col-xs-6 col-sm-6 col-
lg-6 col-md-6'><b>Confirm your details</b></div>" +

```



```

        "</div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'><span id='name'> Name: </span>" +
        "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='nameValue'>" + input.value + "</span> </div></div>" +
        "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-
6'><span id='email'> Email: </span>" +
        "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='emailValue'>" + email.value +
        "</span></div></div><div class='row'><div class='col-xs-6 col-
sm-6 col-lg-6 col-md-6'>" +
        "<span id='Contact'> Contact number: </span></div><div
class='col-xs-6 col-sm-6 col-lg-6 col-md-6'>" +
        "<span id='contactValue'>" + contact.value + "
</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'>" +
        "<span id='Address'> Address: </span> </div><div class='col-xs-6
col-sm-6 col-lg-6 col-md-6'><span id='AddressValue'>" + address.value +
        "</span></div></div>"
        return template;
    }
</script>
<style>
    .control-section {
        padding-left: 10px;
    }
    #target {
        height: 100%;
        min-height: 350px;
    }
    #model_dialog {
        margin: 20px;
    }
    .row {
        padding: 10px 3px;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
    // GET: RTL
    public ActionResult Index()
    {
        ViewBag.DialogButtons = new ButtonModel() { isPrimary = true,
content = "Submit" };
        ViewBag.SubmitButtons1 = new ButtonModel() { content = "YES",
isPrimary = true };
        ViewBag.SubmitButtons2 = new ButtonModel() { content = "NO" };
        return View();
    }
}

```

```
}

```

Customize the Dialog appearance

You can customize the dialog appearance by providing dialog template through [ContentTemplate](#) property. In the following sample, dialog is customized as error window appearance.

CSHTML

```
<div id="target" class="col-lg-12 control-section">
    @Html.EJS().Button("normalbtn").Content("Open").Render()

    @Html.EJS().Dialog("dialog").AnimationSettings(e=>e.Effect(Syncfusion.EJ2.Po
    pups.DialogEffect.Zoom)).Header("File and Folder
    Rename").ShowCloseIcon(true).Width("400px").Target("#target").Buttons(btn=>
    {
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.button1).Add();
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.button2).Add();
    }).ContentTemplate(@<div class='dialog-content'>
        <div class='msg-wrapper col-lg-12'>
            <span class='e-icons close-icon col-lg-2'></span>
            <span class='error-msg col-lg-10'>Can not rename 'pictures'
            because a file or folder with that name already exists </span>
        </div><div class='error-detail col-lg-8'><span>Specify a different
            name</span> </div>
        </div>).Render()
    </div>
    <script>
        document.getElementById('normalbtn').onclick = function () {
            var dialogObj = document.getElementById('dialog').ej2_instances[0];
            dialogObj.show();
        };
        function dlgButtonClick() {
            var dialogObj = document.getElementById('dialog').ej2_instances[0];
            dialogObj.hide();
        }
    </script>
    <style>
        .control-section {
            padding-left: 10px;
        }
        #target {
            height: 100%;
            min-height: 350px;
        }
        span.close-icon {
            width: 24px;
            height: 24px;
            position: relative;
            display: inline-block;
        }
        span.error-msg {
            color: #66afe9;
            display: inline-block;
            position: relative;
        }
        .error-detail.col-lg-8 {

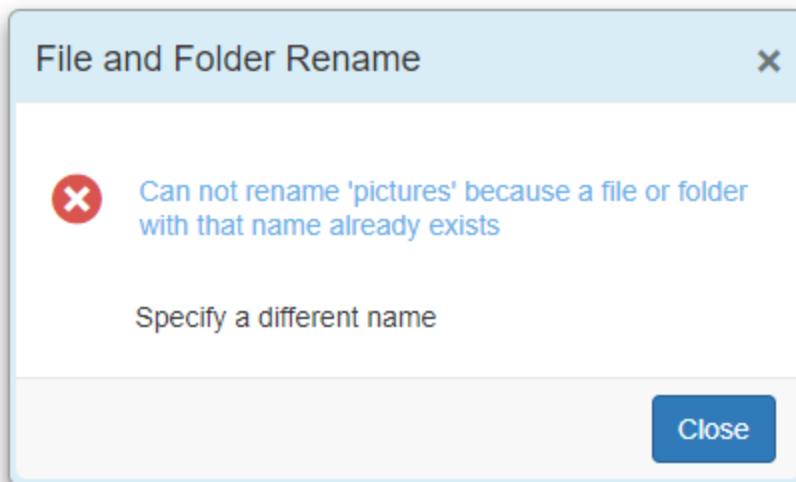
```

```
        position: relative;
        left: 45px;
        margin: 20px 0px 21px;
    }
    .e-icons.close-icon.col-lg-2:before {
        content: '\e7e9';
        font-size: 26px;
        color: #d9534f;
        position: relative;
        left: -12px;
    }
    .e-dialog .e-footer-content {
        background-color: #f8f8f8;
    }
    .e-dialog.e-control.e-popup, .e-dialog.e-control.e-popup .e-dlg-header-
content {
        background-color: #d9edf7;
    }
    .e-dialog.e-control.e-popup {
        padding: 3px;
    }
    .e-dialog.e-control .e-dlg-header-content {
        padding: 10px;
    }
    .e-dialog.e-control .e-footer-content {
        padding: 8px 12px;
    }
    .e-dialog.e-control .e-dlg-content {
        padding: 15px 0px 0px;
    }
}
</style>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.button1 = new { content = "YES", isPrimary = true };
        ViewBag.button2 = new { content = "NO" };
        return View();
    }
}
```

Output be like the below.



Close Dialog when click outside of its region

By default, dialog can be closed by pressing Esc key and clicking the close icon on the right of dialog header. It can also be closed by clicking outside of the dialog using `hide` method. Set the [CloseOnEscape](#) property value to false to prevent closing of the dialog when pressing Esc key.

In the following sample, dialog is closed when clicking outside the dialog area using `hide` method.

CSHTML

```
@using Syncfusion.EJ2.Popups
<div id="target">
    @Html.EJS().Button("normalbtn").Content("Open").Render()

    @Html.EJS().Dialog("dialog").AnimationSettings(e=>e.Effect(DialogEffect.Zoom))
    .Header("Delete Multiple Items").Content("Are you sure you want to permanently delete all of these items?")
    .ShowCloseIcon(true).CloseOnEscape(true).Width("500px").Target("#target").Buttons(btn=> {

        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();

        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
    }).Render()
</div>
<script>
    document.getElementById('normalbtn').onclick = function () {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.show();
    };
    document.getElementById('target').onclick = function (args) {
        if (args.target.tagName != "BUTTON") {
            var dialogObj = document.getElementById('dialog').ej2_instances[0];
            dialogObj.hide();
        }
    };
    function dlgButtonClick() {
```

```

        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
</script>
<style>
    .control-section {
        padding-left: 10px;
    }
    #target {
        height: 100%;
        min-height: 350px;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { content = "YES",
isPrimary = true };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "NO" };
        return View();
    }
}

```

Add Icons to Buttons in Dialog Component

You can add icons to the dialog buttons using the [Buttons](#) property or [FooterTemplate](#) property . For detailed information about dialog buttons, refer to the documentation section.

In the following sample, dialog footer buttons are customized with icons using **Buttons** property.

CSHTML

```

@using Syncfusion.EJ2.Popups
<div id="target" class="col-lg-12 control-section">
    @Html.EJS().Button("normalbtn").Content("Open").Render()

    @Html.EJS().Dialog("dialog").AnimationSettings(a=>a.Effect(DialogEffect.Zoom))
    .Header("Delete Multiple Items").Content("Are you sure you want to permanently delete all of these items?")
    .ShowCloseIcon(true).CloseOnEscape(true).Width("500px").Target("#target").Buttons(btn=> {

        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DefaultButtons1).Add();

        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DefaultButtons2).Add();
    }).Render()
</div>
<script>

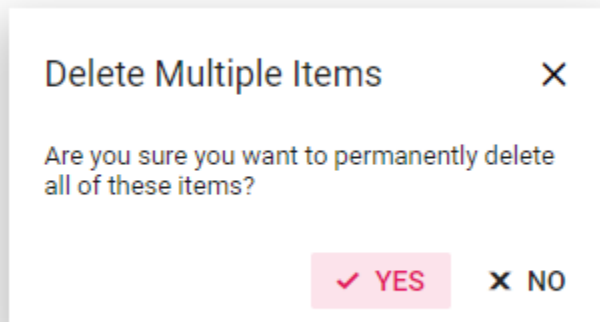
```

```
document.getElementById('normalbtn').onclick = function () {
    var dialogObj = document.getElementById('dialog').ej2_instances[0];
    dialogObj.show();
};
function dlgButtonClick() {
    var dialogObj = document.getElementById('dialog').ej2_instances[0];
    dialogObj.hide();
}
</script>
<style>
    .control-section {
        padding-left: 10px;
    }
    #target {
        height: 100%;
        min-height: 350px;
    }
    .e-ok-icon::before {
        content: '\e7ff';
    }
    .e-close-icon::before {
        content: '\e825';
    }
</style>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
        public string iconCss { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DefaultButtons1 = new ButtonModel() { content = "YES",
isPrimary = true, iconCss = "e-icons e-ok-icon" };
        ViewBag.DefaultButtons2 = new ButtonModel() { content = "NO",
iconCss = "e-icons e-close-icon" };
        return View();
    }
}
```

Output be like the below.



In the following sample, dialog footer buttons are customized with icons using `FooterTemplate` property.

CSHTML

```
@using Syncfusion.EJ2.Popups
<div id="target" class="col-lg-12 control-section">
    @Html.EJS().Button("normalbtn").Content("Open").Render()
    @Html.EJS().Dialog("dialog").Created("onLoad").AnimationSettings(a =>
a.Effect(DialogEffect.Zoom)).Header("Delete Multiple Items").Content("Are
you sure you want to permanently delete all of these
items?").ShowCloseIcon(true).CloseOnEscape(true).Width("500px").Target("#tar
get").FooterTemplate("<div><button id='Button1' class='e-control e-btn e-
primary e-flat' data-ripple='true'><span class='e-btn-icon e-icons e-ok-icon
e-icon-left'></span>Yes</button><button id='Button2' class='e-control e-btn
e-flat' data-ripple='true'><span class='e-btn-icon e-icons e-close-icon e-
icon-left'></span>No</button></div>").Render()
</div>
<script>
    var dialogObj = document.getElementById('dialog').ej2_instances[0];
    document.getElementById('normalbtn').onclick = function () {
        dialogObj.show();
    };
    function onLoad() {
        document.getElementById('Button1').onclick = function () {
            dialogObj.hide();
        };
        document.getElementById('Button2').onclick = function () {
            dialogObj.hide();
        };
    }
</script>
<style>
    .control-section {
        padding-left: 10px;
    }
    #target {
        height: 100%;
        min-height: 350px;
    }
    .e-ok-icon::before {
        content: '\e7ff';
    }
</style>
```

```

    }
    .e-close-icon::before {
        content: '\e825';
    }
</style>

```

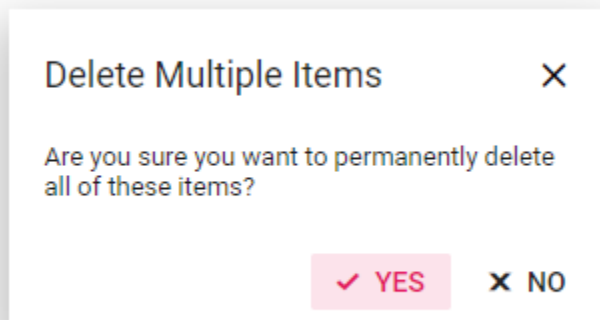
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}

```

Output be like the below.



Setting Maximum Height to the Dialog

By default, the maxHeight for the Dialog is calculated based on the target. If the target is not specified externally, the Dialog consider the body as target and will calculate the maxHeight based on it. We have an option to set the maxHeight of the Dialog in the `beforeOpen` event.

CSHTML

```

<div id="container">
    @Html.EJS().Button("targetButton").Content("Open Dialog").Render()

    @Html.EJS().Dialog("dialog").beforeOpen("onBeforeOpen").Header("Dialog").Visible(false).Content("This is a Dialog with fullscreen display.").Target("#container").Width("800px").Buttons(btn=>
    {
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons1).Add();
        btn.Click("dlgButtonClick").ButtonModel(ViewBag.DialogButtons2).Add();
    }).Render()
</div>
<script>
    document.getElementById('targetButton').onclick = function () {
        var dialog = document.getElementById("dialog").ej2_instances[0];
        dialog.show(true);
    }
</script>

```



```

    }
    function dlgButtonClick() {
        var dialogObj = document.getElementById('dialog').ej2_instances[0];
        dialogObj.hide();
    }
    function onBeforeOpen(args) {
        args.maxHeight = '300px';
    }
</script>
<style>
    #container {
        height: 100%;
        width: 100%;
        position: absolute;
        padding: 20px;
        margin: 0px;
        left: 0px;
        right: 0px;
        bottom: 0px;
        top: 0px;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
        public string cssClass { get; set; }
    }
    public ActionResult Index()
    {
        ViewBag.DialogButtons1 = new ButtonModel() { isPrimary = true,
cssClass = "e-flat", content = "OK" };
        ViewBag.DialogButtons2 = new ButtonModel() { content = "Cancel",
cssClass = "e-flat" };
        return View();
    }
}

```

Migration from Essential JS 1

This article describes the API migration process of Dialog control from Essential JS 1 to Essential JS 2.

Accessibility and localization

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Keyboard Navigation | **Property** : allowKeyboardNavigation

@Html.EJ().Dialog("dialog").AllowKeyboardNavigation(true) | No separate Property for
enable/disable keyboard navigation. Its enabled by default. |

| Localization | **Property** : locale

@Html.EJ().Dialog("dialog").Locale("es-ES") | **Property** : locale

@Html.EJS().Dialog("dialog").Locale("es-ES").Render() |

| Right to left | **Property**: enableRTL

@Html.EJ().Dialog("dialog").EnableRtl(true) | **Property**: enableRTL

@Html.EJS().Dialog("dialog").EnableRtl(true).Render() |

Header

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Header Content | **Property** : title

@Html.EJ().Dialog("dialog").Title("EJ1 Dialog header")
 Method : setTitle
 \$(' #dialog').ejDialog('setTitle', 'EJ1 Dialog Header'); | **Property** : header

@Html.EJS().Dialog("dialog").Header("EJ2 Dialog").Render() |

| close button | **Property** : actionButtons

 @ { List<string> actionButtons = new List<string>(); actionButtons.Add("close"); }

@Html.EJ().Dialog("dialog").ActionButtons(actionButtons)
 | **Property** : showCloseIcon

@Html.EJS().Dialog("dialog").ShowCloseIcon(true).Render() |

| Event triggers when click on action buttons | **Event**: actionButtonClick

 @ { List<string> actionButtons = new List<string>(); actionButtons.Add("close"); }

@Html.EJ().Dialog("dialog").ActionButtons(actionButtons).ClientSideEvents(event =>event.ActionButtonClick("playMedia"))

function playMedia(args) {}
 | Not Applicable |

| Minimize | **Property** : actionButtons

 @ { List<string> actionButtons = new List<string>(); actionButtons.Add("minimize"); }

@Html.EJ().Dialog("dialog").ActionButtons(actionButtons)
 | Not Applicable |

| Maximize | **Property** : actionButtons

@ { List<string> actionButtons = new List<string>(); actionButtons.Add("maximize"); }

@Html.EJ().Dialog("dialog").ActionButtons(actionButtons)
 | Not Applicable |

| Collapse /Expand | **Property** : actionButtons
 Method : collapse(), expand ()

@ { List<string> actionButtons = new List<string>(); actionButtons.Add("collapsible"); }

@Html.EJ().Dialog("dialog").ActionButtons(actionButtons)
 \$(' #dialog').ejDialog('collapse');
 \$(' #dialog').ejDialog('expand') | Not Applicable |

| Event triggers when expanding the collapsed dialog | **Event**: expand

@Html.EJ().Dialog("dialog").ClientSideEvents(evt => evt.Expand("expandAction"))

function expandAction(args) {}
 | Not Applicable |

| Event triggers when collapsing the expanded dialog | **Event**: collapse

@Html.EJ().Dialog("dialog").ClientSideEvents(evt => evt.Collapse("collapseAction"))

function collapseAction(args) {}
 | Not Applicable |

| Pin | **Property** : actionButtons

@ { List<string> actionButtons = new List<string>(); actionButtons.Add("pin"); }

@Html.EJ().Dialog("dialog").ActionButtons(actionButtons)
 | Not Applicable |

| Header visibility | **Property:** showHeader

@Html.EJ().Dialog("dialog").ShowHeader(true) | Not Applicable |

| Close on escape key press | **Property :** closeOnEscape

@Html.EJ().Dialog("dialog").CloseOnEscape(true) | **Property :** closeOnEscape

@Html.EJS().Dialog("dialog").CloseOnEscape(true).Render() |

Footer

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Footer Content | **Property :** footerTemplateId

@Html.EJ().Dialog("dialog").FooterTemplateId("sample") | **Property:** footerTemplate

@Html.EJS().Dialog("dialog").FooterTemplate("<button>Submit</button>").Render() |

| Footer action buttons | Not applicable | **Property :** buttons

@Html.EJS().Dialog("dialog").Buttons(ViewBag.DefaultButtons).Render()

public
 ActionResult DefaultFunctionalities() { List<DialogDialogButton> buttons = new
 List<DialogDialogButton>() { }; buttons.Add(new DialogDialogButton() { Click = "dlgButtonClick",
 ButtonModel = new DefaultButtonModel() { content = "Ok" } }); ViewBag.DefaultButtons =
 buttons; return View(); } public class DefaultButtonModel { public string content { get; set;
 }
 |

| Footer visibility | **Property :** showFooter

@Html.EJ().Dialog("dialog").ShowFooter(true)
 | Not Applicable |

Content

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Dialog content | **Method :** setContent

@Html.EJ().Dialog("dialog")

 \$('#dialog').ejDialog('setContent', 'Dialog Content') | **Property :** content

@Html.EJS().Dialog("dialog").Content("Dialog content").Render() |

| Loading content using AJAX request | **Property :** contentType, contentUrl

@Html.EJ().Dialog("dialog").contentType("ajax").ContentUrl("") | Not Applicable |

| Event triggers after the dialog content loaded in DOM | **Event:** contentLoad

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
 evt.ContentLoad("onLoad"))

function onLoad(args) {}
 | Not Applicable |

| Event trigger when fails to load ajax content | **Event:** ajaxError

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
 evt.AjaxError("onAjaxError"))

function onAjaxError(args) {}
 | Not Applicable |

| Event trigger when load ajax content successfully | **Event:** ajaxSuccess

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
 evt.AjaxSuccess("onAjaxSuccess"))

function onAjaxSuccess(args) {}
 | Not
 Applicable |

Animation

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Enabling Animation | **Property** : enableAnimation

@Html.EJ().Dialog("dialog").EnableAnimation(true) | Not Applicable |

| Animation effects | **Property** : animation.show.effect

@Html.EJ().Dialog("dialog").Animation(animate => animate.Show(show =>

show.Effect("slide"))
 | **Property** : animationSettings.effect

@Html.EJS().Dialog("default_dialog").AnimationSettings(new DialogAnimationSettings() { Effect = DialogEffect.Zoom }).Render()
 |

| Animation duration | **Property**: animation.show.duration

@Html.EJ().Dialog("dialog").Animation(animate => animate.Show(show =>

show.Duration(500).Effect("slide"))
 | **Property** : animationSettings.duration

@Html.EJS().Dialog("default_dialog").AnimationSettings(new DialogAnimationSettings() { Effect = DialogEffect.Zoom, DialogEffect.Duration(500) }).Render()
 |

| Animation delay | Not applicable | **Property**: animationSettings.delay

@Html.EJS().Dialog("default_dialog").AnimationSettings(new DialogAnimationSettings() { Effect = DialogEffect.Zoom, DialogEffect.Duration(500), DialogEffect.Delay(500) }).Render()
 |

Draggable and resizing

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Draggable dialog | **Property** : allowDraggable

@Html.EJ().Dialog("dialog").AllowDraggable(true) | **Property** : allowDragging

@Html.EJS().Dialog("dialog").AllowDragging(true).Render() |

| Event triggers when the user drags the dialog | **Event**: drag

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>

evt.Drag("onDrag"))

function onDrag(args) {}
 | **Event**: drag

@Html.EJS().Dialog("dialog").Drag("onDrag").Render()

function onDrag(args) {}
 |

| Event triggers when the start to drag the dialog | **Event**: dragStart

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>

evt.DragStart("onDragStart"))

function onDragStart(args) {}
 | **Event**: dragStart

@Html.EJS().Dialog("dialog").DragStart("onDragStart").Render()

function onDragStart(args) {}
 |

| Event triggers when the stops to drag the dialog | **Event**: dragStop

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>

evt.DragStop("onDragStop"))

function onDragStop(args) {}
 | **Event**:

dragStop

@Html.EJS().Dialog("dialog").DragStop("onDragStop").Render()

function onDragStop(args) {}
 |

| Resizing dialog | **Property** : enableResize

@Html.EJ().Dialog("dialog").EnableResize(true) | Not applicable |

| Event triggers when resizing the dialog | **Event**: resize

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
 evt.Resize("onReSize"))

function onReSize(args) {}
 | Not Applicable |

| Event triggers when starts to resizing the dialog | **Event**:
 resizeStart

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
 evt.ResizeStart("onResizeStart"))

function onResizeStart(args) {}
 | Not Applicable |

| Event triggers when the stops to resizing the dialog | **Event**:
 resizeStop

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
 evt.ResizeStop("onResizeStop"))

function onResizeStop(args) {}
 | Not Applicable |

Target

Behavior	**Property in Essential JS 1**	**Property in Essential JS 2**

| Target element to append dialog in document | **Property** : target

@Html.EJ().Dialog("dialog").Target("#dialogTarget") | **Property**: target

@Html.EJS().Dialog("dialog").Target("#dialogTarget").Render() |

| Element for draggable area | **Property** : containment

@Html.EJ().Dialog("dialog").Containment("#dragArea") | Not applicable |

Position

Behavior	**Property in Essential JS 1**	**Property in Essential JS 2**

| Customizing dialog position using X, Y coordinate values | **Property** : position

@Html.EJ().Dialog("dialog").ClientSideEvents(evt => evt.Position.X(300).Y(100))
 |
Property : position

 @Html.EJS().Dialog("dialog").Position(obj =>
 obj.X("300").Y("100")).Render()
 |

| positioning dialog using position values | Not Applicable | **Property**: position

 @Html.EJS().Dialog("dialog").Position(obj => obj.X("center").Y("center")).Render()
 |

Visibility

Behavior	**Property in Essential JS 1**	**Property in Essential JS 2**

| Render dialog in visible/hidden state | **Property**: showOnInit

@Html.EJ().Dialog("dialog").ShowOnInit(true) | **Property**: visible

@Html.EJS().Dialog("dialog").visible(false).Render() |

Dialog mode

Behavior	**Property in Essential JS 1**	**Property in Essential JS 2**

| Render modal dialog | **Property** : enableModal

@Html.EJ().Dialog("dialog").EnableModal(true) | **Property** : isModal

 @Html.EJS().Dialog("dialog").IsModal(true).Render() |

Tooltip

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Sets the tooltip for dialog buttons | **Property** : tooltip

 @Html.EJ().Dialog("dialog").ClientSideEvents(evt => evt.Tooltip("tooltip"))

function
 tooltip: object { close: 'Exit' }
 | No Separate Property for tooltip. It renders based on locale text.
 |

Control state

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Enable/Disable the control | **Property** : enabled

 @Html.EJ().Dialog("dialog").Enabled(false) | Not Applicable |
 | Enable/ Disable page scrolling | **Property**: backgroundScroll

@Html.EJ().Dialog("dialog").BackgroundScroll(false) | No separate Property for disabling page
 scroll. By default, scrolling prevented for modal dialog |

State maintenance

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Save the model values in local storage or cookies | **Property** : enablePersistence

@Html.EJ().Dialog("dialog").EnablePersistence(true) | **Property** : enablePersistence

@Html.EJS().Dialog("dialog").EnablePersistence(true).Render() |

Common

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Adjusting Height | **Property** : height

@Html.EJ().Dialog("dialog").Height("400") |
Property : height

 @Html.EJS().Dialog("dialog").Height("50%").Render() |
 | Adjusting width | **Property**: width

@Html.EJ().Dialog("dialog").Width("400") | **Property** :
 width

 @Html.EJS().Dialog("dialog").Width("50%").Render() |
 | Adding custom class | **Property**: cssClass

@Html.EJ().Dialog("dialog").cssClass("custom-
 class") | **Property**: cssClass

 @Html.EJS().Dialog("dialog").CssClass("custom-
 class").Render() |
 | Adding zIndex | **Property**: zIndex

 @Html.EJ().Dialog("dialog").ZIndex("2000") |
Property: zIndex

 @Html.EJS().Dialog("dialog").ZIndex("2000").Render() |
 | Maximum height | **Property**: maxHeight

 @Html.EJ().Dialog("dialog").MaxHeight("600") | Not Applicable |

| Maximum width | **Property:** maxWidth

 @Html.EJ().Dialog("dialog").MaxWidth("600")
| Not Applicable |

| Minimum height | **Property:** minHeight

 @Html.EJ().Dialog("dialog").MinHeight("300")
| Not Applicable |

| Minimum width | **Property:** minWidth

 @Html.EJ().Dialog("dialog").MinWidth("300") |
Not Applicable |

| Adding html attributes | **Property:** htmlAttributes

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
evt.HtmlAttributes("htmlAttributes"))

function htmlAttributes: object { class: 'my-
class' }
 | Not Applicable |

| Custom icon in the header | **Property:** faviconCSS

@Html.EJ().Dialog("dialog").FaviconCSS("custom-icon") | Not Applicable |

| Rounded corner appearance | **Property:** showRoundedCorner

@Html.EJ().Dialog("dialog").ShowRoundedCorner(true) | Not Applicable |

| Make control flexible for mobile view | **Property:** isResponsive

@Html.EJ().Dialog("dialog").IsResponsive(true) | Not Applicable |

| Close the Dialog | **Method:** close()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('close') | **Method :** hide()

@Html.EJS().Dialog("dialog").Render()

var dialogObj=
document.getElementById("dialog").ej2Instances[0]; dialogObj.hide();
 |

| Event triggers before the dialog closes | **Event:** beforeClose

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
evt.BeforeClose("onBeforeClose"))

function onBeforeClose(args) {}
 | **Event:**
beforeClose

@Html.EJS().Dialog("dialog").BeforeClose("onBeforeClose").Render()

function
onBeforeClose(args) {}
 |

| Event triggers when the dialog closes | **Event:** close

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
evt.Close("onClose"))

function onClose(args) {}
 | **Event:** close

@Html.EJS().Dialog("dialog").Close("onClose").Render()

function
onClose(args) {}
 |

| Destroy the control | **Method:** destroy()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('destroy') | **Method:** destroy()

@Html.EJS().Dialog("dialog").Render()

var dialogObj=
document.getElementById("dialog").ej2Instances[0]; dialogObj.destroy();
 |

| Focus the dialog element | **Method:** focus()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('focus') | Not Applicable |

| Check whether the dialog is open | **Method:** isOpen()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('isOpen') | Not Applicable |

| Maximize the dialog | **Method:** maximize()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('maximize') | Not Applicable |

| Minimize the dialog | **Method:** minimize()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('minimize') | Not Applicable |

| Open the dialog | **Method:** open()

 @Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('open') | **Method :** show()

@Html.EJS().Dialog("dialog").Render()

var dialogObj=
document.getElementById("dialog").ej2Instances[0]; dialogObj.show();
 |

| Event trigger before the dialog opens | **Event:** beforeOpen

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
evt.BeforeOpen("onBeforeOpen"))

function onBeforeOpen(args) {}
 | **Event:**
beforeOpen

@Html.EJS().Dialog("dialog").BeforeOpen("onBeforeOpen").Render()

functi
on onBeforeOpen(args) {}
 |

| Event triggers when the opens the dialog | **Event:** open

@Html.EJ().Dialog("dialog").ClientSideEvents(evt => evt.Open("onOpen"))

function
onOpen(args) {}
 | **Event:** open

@Html.EJS().Dialog("dialog").Open("onOpen").Render()

function
onOpen(args) {}
 |

| Refresh the dialog | **Method:** refresh()

@Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('refresh') | **Method :** refreshPosition()

@Html.EJS().Dialog("dialog").Render()

var dialogObj=
document.getElementById("dialog").ej2Instances[0]; dialogObj.refreshPosition();
 |

| Pin/ unpin the dialog | **Method:** pin

@Html.EJ().Dialog("dialog")

\$('#dialog').ejDialog('pin');
\$('#dialog').ejDialog('unpin'); | **Not Applicable** |

| Event triggers after the dialog created successfully | **Event:** create

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
evt.Create("onCreate"))

function onCreate(args) {}
 | **Event :** created

@Html.EJS().Dialog("dialog").Created("onCreated").Render()

function
onCreated(args) {}
 |

| Event triggers when the control destroyed successfully | **Event:** destroy

@Html.EJ().Dialog("dialog").ClientSideEvents(evt =>
evt.Destroy("onDestroy"))

function onDestroy(args) {}
 | Not Applicable |

| Event triggers on clicking on modal dialog overlay | **Not Applicable** | **Event :** overlayClick

@Html.EJS().Dialog("dialog").OverlayClick("onOverlayClick").Render()

functi
on onOverlayClick(args) {}
 |

Document Editor

Overview

The Document Editor component is used to create, edit, view, and print Word documents in web applications. All the user interactions and editing operations that run purely in the client-side provides much faster editing experience to the users.

Key Features

- [Opens](#) the native Syncfusion Document Text (*.sfdt) format documents in the client-side.
- [Saves the documents](#) in the client-side as Syncfusion Document Text (.sfdt) and Word document (.docx).
- Supports document elements like text, [image](#), [table](#), fields, [bookmark](#), [shapes](#), [section](#), [header and footer](#).
- Supports the commonly used fields like [hyperlink](#), page number, page count, and table of contents.
- Supports formats like [text](#), [paragraph](#), [bullets and numbering](#), [table](#), [page settings](#), etc.
- Provides support to create, edit, and apply [paragraph and character styles](#).
- Provides support to [find and replace](#) text within the document.
- Supports all the common editing and formatting operations along with [undo and redo](#).
- Provides support to [cut](#), [copy](#), and [paste](#) rich text contents within the component. Also allows pasting simple text to and from other applications.
- Provides support to insert, and edit [form fields](#).
- Provides support to insert, and edit [comments](#).
- Provides support to track the inserted and deleted content.
- Provides support to perform [spell checking](#) for any input text.
- Allows user interactions like [zoom](#), [scroll](#), select contents through touch, mouse, and keyboard.
- Provides intuitive UI options like context menu, [dialogs](#), and [navigation pane](#).
- [Localizes](#) all the static text to any desired language.
- Allows to create a lightweight Word viewer using module injection to view and [prints](#) Word documents.
- Provides a server-side helper library to open the Word documents like DOCX, DOC, WordML, RTF, and Text, by converting it to SFDT file format.

Supported Web platforms

- [Javascript\(ES5\)](#)
- [Javascript](#)
- [Angular](#)
- [React](#)
- [Vue](#)
- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Blazor](#)

Getting Started with ASP.NET MVC DocumentEditor Control

This section briefly explains about how to include [ASP.NET MVC DocumentEditor](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC DocumentEditor control

Now, add the Syncfusion ASP.NET MVC DocumentEditor control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().DocumentEditor("container").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DocumentEditor control will be rendered in the default web browser.

Note: Starting from v19.3.0.x, we have optimized the accuracy of text size measurements such as to match Microsoft Word pagination for most Word documents. This improvement is included as default behavior along with an optional API [to disable it and retain the document pagination behavior of older versions](#).

Note: [View Sample in GitHub](#).

See also

- [How to localize the Documenteditor container.](#)
- [How to load the document by default.](#)
- [How to customize tool bar.](#)
- [How to resize Document editor component.](#)

Feature modules

Document editor features are segregated into individual feature-wise modules to enable selective referencing. By default, the document editor displays the document in read-only mode. The required modules should be injected to extend its functionality. The following are the selective modules of document editor that can be included as required:

- **Print** - Prints the document.
- **SfdtExport** - Exports the document as Syncfusion Document Text (.SFDT) file.
- **Selection** - Selects a portion of the document and copy it to the clipboard.
- **Search** - Searches specific text and navigate between the results.

- **WordExport** - Exports the document as Word Document (.DOCX) file.
- **TextExport** - Exports the document as Text Document (.TXT) file.
- **Editor** - Performs all kind of editing operations.
- **EditorHistory** - Maintains the history of editing operations so that you can perform undo and redo at any time.
- User interface options such as context menu, options pane, image resizer, and dialog are available as individual modules.

Note: In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a document editor instance.

Refer to the following table.

Module	Property to Enable the functionality for a document editor instance
---	---
Print	<code>@Html.EJS().DocumentEditor("container").EnablePrint(true).Render()</code>
SfdtExport	<code>@Html.EJS().DocumentEditor("container").EnableSfdtExport(true).Render()</code>
Selection	<code>@Html.EJS().DocumentEditor("container").EnableSelection(true).Render()</code>
Search	<code>@Html.EJS().DocumentEditor("container").EnableSearch(true).Render()</code>
WordExport	<code>@Html.EJS().DocumentEditor("container").EnableWordExport(true).Render()</code>
TextExport	<code>@Html.EJS().DocumentEditor("container").EnableTextExport(true).Render()</code>
Editor	<code>@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).Render()</code>
EditorHistory	<code>@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableEditorHistory(true).Render()</code>
OptionsPane(Find)	<code>@Html.EJS().DocumentEditor("container").EnableSearch(true).EnableOptionsPane(true).Render()</code>
OptionsPane(Find and Replace)	<code>@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableSearch(true).EnableOptionsPane(true).Render()</code>
ContextMenu	<code>@Html.EJS().DocumentEditor("container").EnableSelection(true).EnableContextMenu(true).Render()</code>
ImageResizer	<code>@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableImageResizer(true).Render()</code>
HyperlinkDialog	<code>@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableHyperlinkDialog(true).Render()</code>
TableDialog	<code>@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableTableDialog(true).Render()</code>

```
|FontDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableFontDialog(true).Render()|
```

```
|ParagraphDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableParagraphDialog(true).Render()|
```

```
|BookmarkDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableBookmarkDialog(true).Render()|
```

```
|PageSetupDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnablePageSetupDialog(true).Render()|
```

```
|TableOfContentsDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableTableOfContentsDialog(true).Render()|
```

```
|ListDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableListDialog(true).Render()|
```

```
|TablePropertiesDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableTablePropertiesDialog(true).Render()|
```

```
|CellOptionsDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableTablePropertiesDialog(true).Render()|
```

```
|BordersAndShadingDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableBordersAndShadingDialog(true).Render()|
```

```
|TableOptionsDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableTableOptionsDialog(true).Render()|
```

```
|StylesDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableStyleDialog(true).EnableStylesDialog(true).Render()|
```

```
|StyleDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableStyleDialog(true).Render()|
```

```
|BulletsAndNumberingDialog| @Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true).EnableStyleDialog(true).Render()|
```

Import

In Document Editor, the documents are stored in its own format called **Syncfusion Document Text (SFDT)**.

The following example shows how to open SFDT data in Document Editor.

CSHTML

```
@Html.EJS().DocumentEditor("container").Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var sfdt = {
            "sections": [
```

```

        {
            "blocks": [
                {
                    "inlines": [
                        {
                            "characterFormat": {
                                "bold": true,
                                "italic": true
                            },
                            "text": "Hello World"
                        }
                    ]
                }
            ],
            "headersFooters": {
            }
        }
    ];
    documenteditor.open(JSON.stringify(sfdt));
});
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

Import document from local machine

The following example shows how to import document from local machine.

CSHTML

```

<input type="file" id="file_upload" accept=".sfdt" style="position:fixed;
left:-100em" />
@Html.EJS().Button("import").Content("Import").Render()
<div id="documenteditor" style="width:100%;height:100%">
    @Html.EJS().DocumentEditor("container").Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        document.getElementById("import").addEventListener("click", function
    () {
        document.getElementById('file_upload').click();
    });
    document.getElementById('file_upload').addEventListener("change",
function (e) {
    if (e.target.files[0]) {
        var file = e.target.files[0];

```

```

        if (file.name.substr(file.name.lastIndexOf('.')) ===
'.sfdt') {
            var fileReader = new FileReader();
            fileReader.onload = function (e) {
                var contents = e.target.result;
                documenteditor.open(contents);
            };
            fileReader.readAsText(file);
            documenteditor.documentName = file.name.substr(0,
file.name.lastIndexOf('.'));
        }
    });
});
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

Convert word documents into SFDT

You can convert word documents into SFDT format using the [SynCFusion.EJ2.WordEditor.AspNet.MVC4](#) or [SynCFusion.EJ2.WordEditor.AspNet.MVC5](#) by the web API service implementation. This library helps you to convert word documents (.dotx,.docx,.docm,.dot,.doc), rich text format documents (.rtf), and text documents (.txt) into SFDT format. Refer to the following example.

CSHTML

```

<input type="file" id="file_upload" style="position:fixed; left:-100em" />
@Html.EJS().Button("import").Content("Import").Render()
<div id="documenteditor" style="width:100%;height:100%">
    @Html.EJS().DocumentEditor("container").Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        document.getElementById('file_upload').setAttribute('accept',
'.dotx,.docx,.docm,.dot,.doc,.rtf,.txt,.xml,.sfdt');
        document.getElementById('file_upload').addEventListener("change",
function (e) {
            if (e.target.files[0]) {
                var file = e.target.files[0];
                if (file.name.substr(file.name.lastIndexOf('.')) !==
'.sfdt') {
                    loadFile(file);
                }
            }
        });
    });

```

```
function loadFile(file) {
    var ajax = new XMLHttpRequest();
    ajax.open('POST',
'https://localhost:4000/api/documenteditor/Import', true);
    ajax.onreadystatechange = function () {
        if (ajax.readyState === 4) {
            if (ajax.status === 200 || ajax.status === 304) {
                // open SFDT text in document editor
                documenteditor.open(ajax.responseText);
            }
        }
    }
    var formData = new FormData();
    formData.append('files', file);
    ajax.send(formData);
}
});
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Here's how to handle the server-side action for converting word document in to SFDT.

```
`csharp
[HttpPost]
public HttpResponseMessage Import()
{
    if (HttpContext.Current.Request.Files.Count == 0)
        return null;
    HttpPostedFile file = HttpContext.Current.Request.Files[0];
    int index = file.FileName.LastIndexOf('.');
    string type = index > -1 && index < file.FileName.Length - 1 ?
file.FileName.Substring(index) : ".docx";
    Stream stream = file.InputStream;
    stream.Position = 0;
    EJ2WordDocument document = EJ2WordDocument.Load(stream, GetFormatType(type.ToLower()));
    string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
    document.Dispose();
    stream.Close();
}
```



```
return new HttpResponseMessage() { Content = new StringContent(json) };
}
internal static FormatType GetFormatType(string format)
{
    if (string.IsNullOrEmpty(format))
        throw new NotSupportedException("EJ2 DocumentEditor does not support this file format.");
    switch (format.ToLower()) {
        case ".dotx":
        case ".docx":
        case ".docm":
        case ".dotm":
            return FormatType.Docx;
        case ".dot":
        case ".doc":
            return FormatType.Doc;
        case ".rtf":
            return FormatType.Rtf;
        case ".txt":
            return FormatType.Txt;
        case ".xml":
            return FormatType.WordML;
        default:
            throw new NotSupportedException("EJ2 DocumentEditor does not support this file format.");
    }
}
`
```

[See Also](#)

- [Feature modules](#)

Export in Document Editor Component

Document editor exports the document into various known file formats in client-side such as Microsoft Word document (.docx), text document (.txt), and its own format called **Syncfusion Document Text (.sfdt)**.

We are providing two types of save APIs as mentioned below.

API name	Purpose
save(filename,FormatType):void	FormatType: Sfdt or Docx or Txt Creates the document with specified file name and format type. Then, the created file is downloaded in the client browser by default.
saveAsBlob(FormatType):Blob	Creates the document in specified format type and returns the created document as Blob. This blob can be uploaded to your required server, database, or file path.

Sfdt export

The following example shows how to export documents in document editor as Syncfusion document text (.sfdt).

CSHTML

```
@Html.EJS().Button("export").Content("Export").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        document.getElementById('export').addEventListener('click', function
    () {
        documenteditor.save('sample', 'Sfdt');
    });
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Note: To enable Sfdt export for a document editor instance, set [enableSfdtExport](#) to true.

Word export

The following example shows how to export the document as Word document (.docx).

CSHTML

```
@Html.EJS().Button("export").Content("Export").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
```

```
.EnableSelection(true).EnableSfdtExport(true).EnableWordExport(true).Render(
)
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        document.getElementById('export').addEventListener('click', function
    () {
        documenteditor.save('sample', 'Docx');
    });
});
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Note: To enable word export for a document editor instance, set [enableWordExport](#) to true.

Text export

The following example shows how to export document as text document (.txt).

CSHTML

```
@Html.EJS().Button("export").Content("Export").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).EnableTextExport(true).Render(
)
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        document.getElementById('export').addEventListener('click', function
    () {
        documenteditor.save('sample', 'Txt');
    });
});
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
```

```
return View();
}
```

Note: To enable text export for a document editor instance, set [enableTextExport](#) to true.

Export as blob

Document editor also supports API to store the document into a blob.

CSHTML

```
@Html.EJS().Button("export").Content("Export").Render()
<div id="documenteditor">

@Html.EJS().DocumentEditor("container").EnableSfdtExport(true).EnableWordExport(true).EnableTextExport(true).Render()
</div>
<script>
    var documenteditor;
    documenteditor = documenteditorElement.ej2_instances[0];
    documenteditor.resize();
    documenteditor.open(sfdt);
    document.getElementById('export').addEventListener('click', function ()
    {
        documenteditor.saveAsBlob('Docx').then(function (exportedDocument) {
            // The blob can be processed further
        });
    });
</script>
```

EXPORT-BLOB.CS

For instance, to export the document as Rich Text Format file, implement an ASP.NET MVC web API controller using DocIO library by passing the DOCX blob.

```
`csharp
```

```
//API controller for the conversion.
```

```
[HttpPost]
```

```
public HttpResponseMessage ExportAsRtf()
```

```
{
```

```
System.Web.HttpPostedFile data = HttpContext.Current.Request.Files[0];
```

```
//Opens document stream
```

```
WordDocument wordDocument = new WordDocument(data.InputStream);
```

```
MemoryStream stream = new MemoryStream();
```

```
//Converts document stream as RTF
```

```
wordDocument.Save(stream, FormatType.Rtf);
wordDocument.Close();
stream.Position = 0;
return new HttpResponseMessage() { Content = new StreamContent(stream) };
}
,
```

In client-side, you can consume this web service and save the document as Rich Text Format (.rtf) file.

CSHTML

```
document.getElementById('export').addEventListener('click', () => {
    documenteditor.saveAsBlob('Docx').then(function (exportedDocument) {
        // The blob can be processed further
        var formData = new FormData();
        formData.append('fileName', 'sample.docx');
        formData.append('data', exportedDocument);
        saveAsRtf(formData);
    });
});
function saveAsRtf(formData) {
    var httpRequest = new XMLHttpRequest();
    httpRequest.open('POST', '/api/DocumentEditor/ExportAsRtf', true);
    httpRequest.onreadystatechange = function () {
        if (httpRequest.readyState === 4) {
            if (httpRequest.status === 200 || httpRequest.status === 304) {
                if (!!navigator.msSaveBlob) {
                    navigator.msSaveBlob(httpRequest.response,
'sample.rtf');
                } else {
                    var downloadLink =
document.createElementNS('http://www.w3.org/1999/xhtml', 'a');
                    download('sample.rtf', 'rtf', httpRequest.response,
downloadLink, 'download' in downloadLink);
                }
            } else {
                console.error(httpRequest.response);
            }
        }
    }
    httpRequest.responseType = 'blob';
    httpRequest.send(formData);
}
function download(fileName, extension, buffer, downloadLink,
hasDownloadAttribute) {
    if (hasDownloadAttribute) {
        downloadLink.download = fileName;
        var dataUrl = window.URL.createObjectURL(buffer);
        downloadLink.href = dataUrl;
        var event = document.createEvent('MouseEvent');
        event.initEvent('click', true, true);
        downloadLink.dispatchEvent(event);
        setTimeout(function () {
            window.URL.revokeObjectURL(dataUrl);
        }, 1000);
    }
}
```

```

        dataUrl = undefined;
    });
    } else {
        if (extension !== 'docx' && extension !== 'xlsx') {
            var url = window.URL.createObjectURL(buffer);
            var isPopupBlocked = window.open(url, '_blank');
            if (!isPopupBlocked) {
                window.location.href = url;
            }
        }
    }
}

```

EXPORT-RTF.CS

See Also

- [Feature modules](#)
- [How to export the document as pdf.](#)

Collaborative Editing (preview)

Allows multiple users to work on the same document simultaneously. This can be done in real-time, so that collaborators can see the changes as they are made. Collaborative editing can be a great way to improve efficiency, as it allows team members to work together on a document without having to wait for others to finish their changes.

Note: Collaborative editing support is currently in preview mode only and is not yet ready for production environments.

Prerequisites

Following things are needed to enable collaborative editing in Document Editor

- SignalR
- Microsoft SQL Server

How to enable collaborative editing in client side

Step 1: Enable collaborative editing in Document Editor

To enable collaborative editing, inject `CollaborativeEditingHandler` and set the property `enableCollaborativeEditing` to true in the Document Editor, like in the code snippet below.

CSHTML

```

<div>
    <div id='documenteditor_titlebar' class="e-de-ctn-title"></div>
    @Html.EJS().DocumentEditorContainer("container").Height("590px").EnableToolbar(true).Render()
</div>
<script>
    container.documentEditor.enableCollaborativeEditing = true;

```

```
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Step 2: Configure SignalR to send and receive changes

To broadcast the changes made and receive changes from remote users, configure SignalR like below.

CSHTML

```
<script>
    var connection = new signalR.HubConnectionBuilder().withUrl(serviceUrl +
    '/documenteditorhub', {
        skipNegotiation: true,
        transport: signalR.HttpTransportType.WebSockets
    }).withAutomaticReconnect().build();
    connection.onclose(async () => {
        if (connection.state === signalR.HubConnectionState.Disconnected) {
            alert('Connection lost. Please reload the browser to continue.');
        }
    });
    async function start(data) {
        try {
            connection.start().then(function () {
                connection.send('JoinGroup', { roomName: data.roomName,
                currentUser: data.currentUser });
                console.log('server connected!!!');
            });
        } catch (err) {
            console.log(err);
            setTimeout(start, 5000);
        }
    };
    connection.on('dataReceived', onDataReceived.bind(this));
    function onDataReceived(action, data) {
        if (connections) {
            if (action == 'connectionId') {
                connectionId = data;
            } else if (connectionId != data.connectionId) {
                if (action == 'action' || action == 'addUser') {
                    titleBar.addUser(data);
                } else if (action == 'removeUser') {
                    titleBar.removeUser(data);
                }
            }
            connections.applyAction(action, data);
        }
    }
}
</script>
```

Step 3: Join SignalR room while opening the document

When opening a document, we need to generate a unique ID for each document. These unique IDs are then used to create rooms using SignalR, which facilitates sending and receiving data from the server.

CSHTML

```
<script>
    var connectionId = '';
    let serviceUrl = window.location.origin;
    let connections;

    function loadDefaultDocument() {
        const queryString = window.location.search;
        const urlParams = new URLSearchParams(queryString);
        let roomId = urlParams.get('id');
        if (roomId == null) {
            roomId = Math.random().toString(32).slice(2)
            window.history.replaceState({}, "", `?id=` + roomId);
        }
        ejs.popups.showSpinner(document.getElementById('editor_area'));
        var httpRequest = new XMLHttpRequest();
        httpRequest.open('Post', '/api/CollaborativeEditing/ImportFile',
true);
        httpRequest.setRequestHeader("Content-Type",
"application/json;charset=UTF-8");
        httpRequest.onreadystatechange = function () {
            if (httpRequest.readyState === 4) {
                if (httpRequest.status === 200 || httpRequest.status ===
304) {
                    var data = JSON.parse(httpRequest.responseText);
                    connections =
container.documentEditor.collaborativeEditingHandlerModule;
                    connections.updateRoomInfo(roomId, data.version,
serviceUrl + '/');
                    container.documentEditor.open(data.sfdt);
                    setTimeout(function () {
                        start({ action: 'connect', roomName: roomId,
currentUser: container.currentUser }, null);
                    });
                }
            }
        };
        ejs.popups.hideSpinner(document.getElementById('editor_area'));
        tooltip.open();
    }
    else {
        ejs.popups.hideSpinner(document.getElementById('editor_area'));
        alert('Fail to load the document');
    }
}

container.documentEditor.documentName = 'Giant Panda';
httpRequest.send(JSON.stringify({ "fileName": 'Giant Panda.docx',
"roomName": roomId }));
titleBar.updateDocumentTitle();
}
</script>
```


Step 4: Broadcast current editing changes to remote users

Changes made on the client-side need to be sent to the server-side to broadcast them to other connected users. To send the changes made to the server, use the method shown below from the document editor using the `contentChange` event.

CSHTML

```
<script>
    container.contentChange = function (args) {
        if (connections) {
            connections.sendActionToServer(args.operations);
        }
    }
</script>
```

How to enable collaborative editing in ASP.NET Core

Step 1: Configure SignalR in ASP.NET Core

We are using Microsoft SignalR to broadcast the changes. Please add the following configuration to your application's Program.cs file.

```
`csharp
using Microsoft.Azure.SignalR;

.....

builder.Services.AddSignalR();

.....

.....

.....

app.MapHub<DocumentEditorHub>("/documenteditorhub");

.....

.....

`
```

Step 2: Configure SignalR hub to create room for collaborative editing session

To manage groups for each document, create a folder named "Hub" and add a file named "DocumentEditorHub.cs" inside it. Add the following code to the file to manage SignalR groups using room names.

Join the group by using unique id of the document by using `JoinGroup` method.

```
`csharp
static Dictionary<string, ActionInfo> userManager = new Dictionary<string, ActionInfo>();
internal static Dictionary<string, List<ActionInfo>> groupManager = new Dictionary<string,
List<ActionInfo>>();

// Join to the specified room name
public async Task JoinGroup(ActionInfo info)
```

```
{
if (!userManager.ContainsKey(Context.ConnectionId))
{
userManager.Add(Context.ConnectionId, info);
}
info.ConnectionId = Context.ConnectionId;
//Add the current connected use to the specified group
await Groups.AddToGroupAsync(Context.ConnectionId, info.RoomName);
if (groupManager.ContainsKey(info.RoomName))
{
await Clients.Caller.SendAsync("dataReceived", "addUser", groupManager[info.RoomName]);
}
lock (groupManager)
{
if (groupManager.ContainsKey(info.RoomName))
{
groupManager[info.RoomName].Add(info);
}
else
{
List<ActionInfo> actions = new List<ActionInfo>
{
info
};
groupManager.Add(info.RoomName, actions);
}
}
// Notify other users in the group about new user joined the collaborative editing session.
Clients.GroupExcept(info.RoomName, Context.ConnectionId).SendAsync("dataReceived", "addUser",
info);
}
```

Handle user disconnection using SignalR.

```

`csharp
//Handle disconnection from group.
public override Task OnDisconnectedAsync(Exception? e)
{
    string roomName = userManager[Context.ConnectionId].RoomName;
    if (groupManager.ContainsKey(roomName))
    {
        groupManager[roomName].Remove(userManager[Context.ConnectionId]);
        if (groupManager[roomName].Count == 0)
        {
            groupManager.Remove(roomName);
            //If all user disconnected from current room. Auto save the change to source document.
            CollaborativeEditingController.UpdateOperationsToSourceDocument(roomName,
            "<<documentpath>>", false);
        }
    }
    if (userManager.ContainsKey(Context.ConnectionId))
    {
        //Notify other user in the group about user exit the collaborative editing session
        Clients.OthersInGroup(roomName).SendAsync("dataReceived", "removeUser", Context.ConnectionId);
        Groups.RemoveFromGroupAsync(Context.ConnectionId, roomName);
        userManager.Remove(Context.ConnectionId);
    }
    return base.OnDisconnectedAsync(e);
}
`

```

Step 3: Configure Microsoft SQL database connection string in application level

Configure the SQL database that stores temporary data for the collaborative editing session. Provide the SQL database connection string in `appsettings.json` file.

```

`json
.....
"ConnectionStrings": {
"DocumentEditorDatabase": "<SQL server connection string>"
}

```

.....
,

Step 4: Configure Web API actions for collaborative editing

Import File

1. When opening a document, create a database table to store temporary data for the collaborative editing session. 2. If the table already exists, retrieve the records from the table and apply them to the WordDocument instance using the `UpdateActions` method before converting it to the SFDT format.

```
`csharp
public string ImportFile([FromBody] FileInfo param)
{
    .....
    .....
    DocumentContent content = new DocumentContent();
    .....
    //Get source document from database/file system/blob storage
    WordDocument document = GetDocumentFromDatabase(param.fileName, param.documentOwner);
    .....
    //Get temporary records from database
    List<ActionInfo> actions = CreatedTable(param.fileName);
    if(actions!=null)
    {
        //Apply temporary data to the document.
        document.UpdateActions(actions);
    }
    string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
    content.version = 0;
    content.sfdt = json;
    return Newtonsoft.Json.JsonConvert.SerializeObject(content);
}
,
```

Update editing records to database.

Each edit operation made by the user is sent to the server and is pushed to the database. Each operation receives a version number after being inserted into the database.

After inserting the records to the server, the position of the current editing operation must be transformed against any previous editing operations not yet synced with the client using the TransformOperation method.

After performing the transformation, the current operation is broadcast to all connected users within the group.

```
`csharp
```

```
public async Task<ActionInfo> UpdateAction([FromBody] ActionInfo param)
```

```
{
```

```
try
```

```
{
```

```
    ActionInfo modifiedAction = AddOperationsToTable(param);
```

```
    //After transformation broadcast changes to all users in the group
```

```
    await _hubContext.Clients.Group(param.RoomName).SendAsync("dataReceived", "action",  
    modifiedAction);
```

```
    return modifiedAction;
```

```
}
```

```
catch
```

```
{
```

```
    return null;
```

```
}
```

```
}
```

```
private ActionInfo AddOperationsToTable(ActionInfo action)
```

```
{
```

```
    int clientVersion = action.Version;
```

```
    string tableName = action.RoomName;
```

```
    .....
```

```
    .....
```

```
    .....
```

```
    .....
```

```
    List<ActionInfo> actions = GetOperationsQueue(tableName);
```

```
    foreach (ActionInfo info in actions)
```

```
    {
```

```
        if (!info.IsTransformed)
```

```
        {
```

```
CollaborativeEditingHandler.TransformOperation(info, actions);
}
}
action = actions[actions.Count - 1];
action.Version = updateVersion;
//Return the transformed operation to broadcast it to other clients.
return action;
}
`
```

[Add Web API to get previous operation as a backup to get lost operations](#)

On the client side, messages broadcasted using SignalR may be received in a different order, or some operations may be missed due to network issues. In these cases, we need a backup method to retrieve missing records from the database.

Using the following method, we can retrieve all operations after the last successful client-synced version and return all missing operations to the requesting client.

```
`csharp
public async Task<ActionInfo> UpdateAction([FromBody] ActionInfo param)
{
    try
    {
        ActionInfo modifiedAction = AddOperationsToTable(param);
        //After transformation broadcast changes to all users in the group
        await _hubContext.Clients.Group(param.RoomName).SendAsync("dataReceived", "action",
            modifiedAction);
        return modifiedAction;
    }
    catch
    {
        return null;
    }
}

private ActionInfo AddOperationsToTable(ActionInfo action)
{
    int clientVersion = action.Version;
    string tableName = action.RoomName;
```

```

.....
.....
.....
.....
List<ActionInfo> actions = GetOperationsQueue(table);
foreach (ActionInfo info in actions)
{
    if (!info.IsTransformed)
    {
        CollaborativeEditingHandler.TransformOperation(info, actions);
    }
}
action = actions[actions.Count - 1];
action.Version = updateVersion;
//Return the transformed operation to broadcast it to other clients.
return action;
}
`

```

Full version of the code discussed about can be found in below GitHub location.

Github Example: [Collaborative editing examples](#)

Images

Document Editor supports common raster format images like PNG, BMP, JPEG, SVG and GIF. You can insert an image file or online image in the document using the `insertImage()` method.

CSHTML

```

<input type="file" id="insertImageButton" style="position:fixed; left:-110em" accept=".jpg,.jpeg,.png,.bmp" />
@Html.EJS().Button("insert_picture").Content("Image").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableImageResizer(true).EnableEditorHistory(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        document.getElementById('insert-picture').addEventListener('click',
function () {

```

```

        var pictureUpload =
document.getElementById("insertImageButton");
        pictureUpload.value = '';
        pictureUpload.click();
    });

document.getElementById('insertImageButton').addEventListener('change',
onInsertImage);
    function onInsertImage(args) {
        if (navigator.userAgent.match('Chrome') ||
navigator.userAgent.match('Firefox') || navigator.userAgent.match('Edge') ||
navigator.userAgent.match('MSIE') || navigator.userAgent.match('.NET')) {
            if (args.target.files[0]) {
                var path = args.target.files[0];
                var reader = new FileReader();
                reader.onload = function (frEvent) {
                    var base64String = frEvent.target.result;
                    var image = document.createElement('img');
                    image.addEventListener('load', function () {
                        documenteditor.editor.insertImage(base64String,
this.width, this.height);
                    })
                    image.src = base64String;
                };
                reader.readAsDataURL(path);
            }
            //Safari does not Support FileReader Class
        } else {
            var image = document.createElement('img');
            image.addEventListener('load', function () {
                documenteditor.editor.insertImage(args.target.value);
            })
            image.src = args.target.value;
        }
    }
});
</script>

```

IMAGE.CS

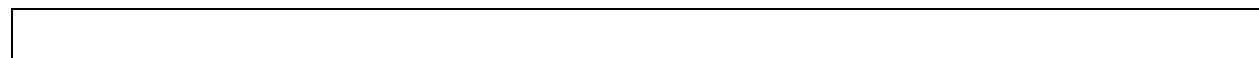


Image files will be internally converted to base64 string. Whereas, online images are preserved as URL.

Image resizing

Document editor provides built-in image resizer that can be injected into your application based on the requirements. This allows to resize the image by dragging the resizing points using mouse or touch interactions. This resizer appears as follows.



Changing size

Document editor exposes API to get or set the size of the selected image.

```
`typescript
```

```
documenteditor.selection.imageFormat.width = 800;
```

```
documenteditor.selection.imageFormat.height = 800;
```

```
,
```

Note: Images are stored and processed (read/write) as base64 string in DocumentEditor. The online image URL is preserved as a URL in DocumentEditor upon saving.

Text wrapping style

Text wrapping refers to how images fit with surrounding text in a document. [Refer to this page](#) for more information about text wrapping styles available in Word documents.

Positioning the image

DocumentEditor preserves the position properties of the image and displays the image based on position properties. It does not support modifying the position properties. Whereas the image will be automatically moved along with text edited if it is positioned relative to the line or paragraph.

See Also

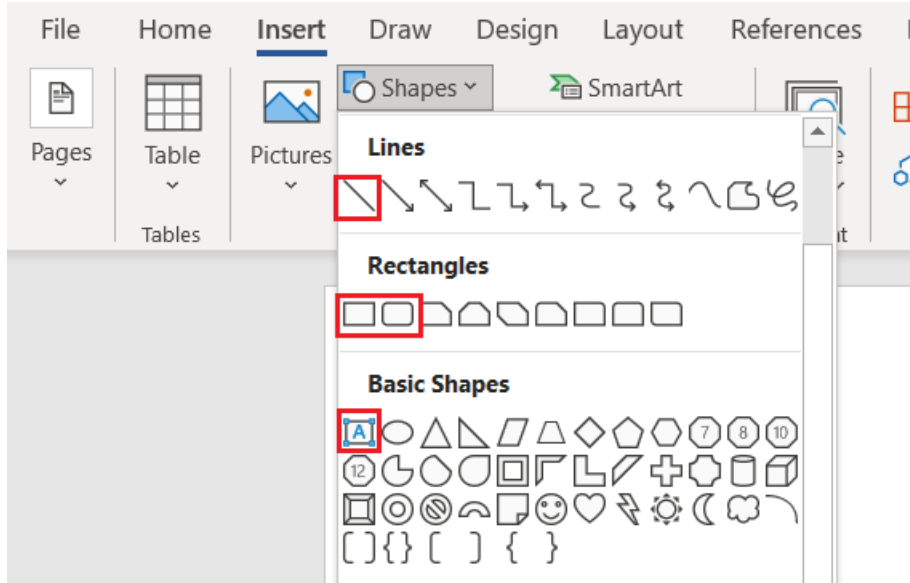
- [Feature modules](#)

Shapes in DocumentEditor

Shapes are drawing objects that include a text box, rectangles, lines, curves, circles, etc. It can be preset or custom geometry. At present, DocumentEditor does not have support to insert shapes. However, if the document contains a shape while importing, it will be preserved properly.

Supported shapes

The DocumentEditor has preservation support for Text box, Rectangle, Rounded Rectangle and Line shapes.



Note: When using ASP.NET MVC service, the unsupported shapes will be converted as image and preserved as image.

Text box Shape

A text box is a rectangular area on the document where you can enter text. When you click in a text box, a flashing cursor will display indicating that you can begin typing. It allows you to enter multiple lines of text with all text formatting.

```

```

Shape Resizer

The DocumentEditor also supports a built-in shape resizer to resize the shapes present in the document. The shape resizer accepts both touch and mouse interactions.



Text wrapping style

Text wrapping refers to how shapes fit with surrounding text in a document. [Refer to this page](#) for more information about text wrapping styles available in Word documents.

Positioning the shape

DocumentEditor preserves the position properties of the shape and displays the shape based on position properties. It does not support modifying the position properties. Whereas the shape will be automatically moved along with text edited if it is positioned relative to the line or paragraph.

Text Wrapping Style in DocumentEditor

Text wrapping refers to how images and shapes are fit with surrounding text in a document. Currently, DocumentEditor has only preservation support for image and textbox shape with below wrapping styles.

In-Line with Text

In this option, the image or shape is placed on the same line surrounding with text like any other word or letter. This image or shape will be automatically moved along with the text while editing, whereas the other options denote that the image or shape stays in a fixed position while the text shifts and wraps around it.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company



manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in

In Front of Text

In this option, the image or shape is placed in front of the text. This can be used to place an image around some text or to add shape to highlight the part in a paragraph.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales centers are located and shipped in throughout their market base.



Note: Starting from v18.2.0.x, the in front of wrapping styles are supported.

Top and Bottom

In this option, Text wraps above and below the image or shape. No text is to the left or right of the image or shape. This can be used for larger images or shapes that occupy most of the width in a document.

Note: Starting from v19.1.0.x, the top and bottom wrapping style is supported.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company

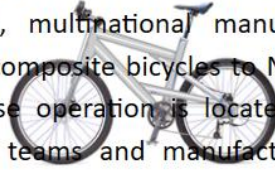


manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290

Behind

In this option, the image or shape is placed behind the text. This can be used when you need to add a watermark or background image to a document.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams and manufacturer and located and shipped in throughout their market base.



Note: Starting from v19.2.0.x, behind text wrapping styles are supported.

Square

In this option, Text wraps around the image or text box in a square shape.

Note: Tight and Through styles will be preserved as square wrapping style in DocumentEditor which is supported from v19.2.0.x.

Adventure Works Cycles, the fictitious company on which the Adventure Works sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams are located throughout their market base.

Adventure works cycles
company.

Bookmarks

Bookmark is a powerful tool that helps to mark a place in the document to find again easily. You can enter many bookmarks in the document and give each one a unique name to identify easily.

Document editor provides built-in dialog to add, delete, and navigate bookmarks within the document. To add a bookmark, select a portion of text in the document. After that, jump to the location or add links to it within the document using built-in hyperlink dialog. You can also delete bookmarks from a document.

Note: Bookmark names need to begin with a letter. They can include both numbers and letters, but not spaces. To separate the words, use an underscore. Bookmark names starting with an underscore are called hidden bookmarks. For example, bookmarks generated for table of contents.

Add bookmark

Using `[insertBookmark]` method, Bookmark can be added to the selected text.

`csharp

```
container.documentEditor.editor.insertBookmark("Bookmark1");
```

`

Select Bookmark

You can select the bookmark in the document using [selectBookmark] method by providing Bookmark name to select as shown in the following code snippet.

```
`csharp
container.documentEditor.selection.selectBookmark("Bookmark1");
`
```

Delete Bookmark

You can delete bookmark in the document using [deleteBookmark] method as shown in the following code snippet.

```
`csharp
container.documentEditor.editor.deleteBookmark("Bookmark1");
`
```

Get Bookmark

You can get all the bookmarks in the document using [getBookmarks] method as shown in the following code snippet.

```
`csharp
container.documentEditor.selection.getBookmarks(false);
`
```

Note: Parameter denotes is include hidden bookmarks. If false, ignore hidden bookmark.

Bookmark Dialog

The following example shows how to open bookmark dialog in Document Editor.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableBookmarkDialog(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    var containerPanel;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open Bookmark Dialog
            documenteditor.showDialog('Bookmark');
        });
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

See Also

- [Feature modules](#)
- [Bookmark dialog](#)

Hyperlink

Document editor supports hyperlink field. You can link a part of the document content to Internet or file location, mail address, or any text within the document.

Navigate a hyperlink

Document editor triggers 'requestNavigate' event whenever user clicks Ctrl key or tap a hyperlink within the document. This event provides necessary details about link type, navigation URL, and local URL (if any) as arguments, and allows to easily customize the hyperlink navigation functionality.

Add the requestNavigate event for DocumentEditor

The following example illustrates how to add requestNavigate event for DocumentEditor.

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").EnableSelection(true).EnableSfd
tExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
        documenteditor.requestNavigate = function (args) {
            if (args.linkType !== 'Bookmark') {
                var link = args.navigationLink;
                if (args.localReference.length > 0) {
                    link += '#' + args.localReference;
                }
                window.open(link);
                args.isHandled = true;
            }
        };
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
```

```
{
    return View();
}
```

Add the requestNavigate event for DocumentEditorContainer component

The following example illustrates how to add requestNavigate event for DocumentEditorContainer component.

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditorContainer("container").EnableToolbar(true).Render(
)
</div>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement;
        var container;
        documenteditor = document.getElementById('container');
        container = documenteditor.ej2_instances[0];
        container.documentEditor.requestNavigate = function (args) {
            if (args.linkType !== 'Bookmark') {
                var link = args.navigationLink;
                if (args.localReference.length > 0) {
                    link += '#' + args.localReference;
                }
                window.open(link);
                args.isHandled = true;
            }
        };
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

If the selection is in hyperlink, trigger this event by calling 'navigateHyperlink' method of 'Selection' instance.

```
`typescript
```

```
documenteditor.selection.navigateHyperlink();
```

```
,
```

[Copy link](#)

Document editor copies link text of a hyperlink field to the clipboard if the selection is in hyperlink.

```
`typescript
```

```
documenteditor .selection.copyHyperlink();
```

Add hyperlink

To create a basic hyperlink in the document, press **ENTER / SPACEBAR / SHIFT + ENTER / TAB** key after typing the address, for instance <http://www.google.com>. Document editor automatically converts this address to a hyperlink field. The text can be considered as a valid URL if it starts with any of the following.

Note: ``http://`
`

`
 `https://`
`

`
 file:///
`

`
 www.
`

`
 mailto:
`

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableEditor(
true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
        documenteditor.requestNavigate = function (args) {
            if (args.linkType !== 'Bookmark') {
                var link = args.navigationLink;
                if (args.localReference.length > 0) {
                    link += '#' + args.localReference;
                }
                window.open(link);
                args.isHandled = true;
            }
        };
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```


Customize screen tip

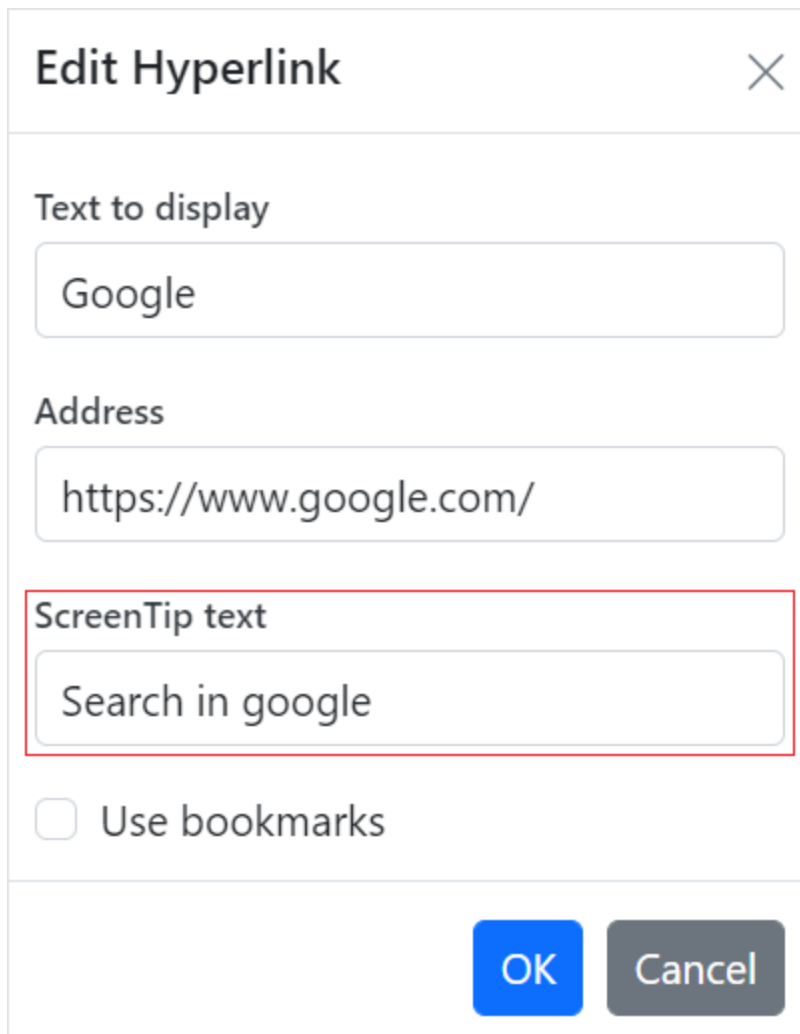
You can customize the screen tip text for the hyperlink by using below sample code.

```
`typescript
```

```
documenteditor.insertHyperlink('https://www.google.com', 'Google', '<<Screen tip text>>');
```

```
,
```

Screen tip text can be modified through UI by using the [Hyperlink dialog](#)



Edit Hyperlink ✕

Text to display

Google

Address

https://www.google.com/

ScreenTip text

Search in google

☐ Use bookmarks

OK Cancel

Remove hyperlink

To remove link from hyperlink in the document, press Backspace key at the end of a hyperlink. By removing the link, it will be converted as plain text. You can use 'removeHyperlink' method of 'Editor' instance if the selection is in hyperlink.

```
`typescript
```

```
documenteditor.editor.removeHyperlink();
```

```
,
```

Hyperlink dialog

Document editor provides dialog support to insert or edit a hyperlink.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableEditor(
true).EnableSelection(true).EnableSfdtExport(true).EnableHyperlinkDialog(tru
e).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
    });
    //Click the Dialog button, the hyperlink dialog will open
    document.getElementById('dialog').addEventListener('click', function ()
{
        documenteditor.showDialog('Hyperlink');
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

You can use the following keyboard shortcut to open the hyperlink dialog if the selection is in hyperlink.

Key Combination	Description
----- -----	
Ctrl + K	Open hyperlink dialog that allows you to create or edit hyperlink

See Also

- [Feature modules](#)
- [Hyperlink dialog](#)

Tables

Tables are an efficient way to present information. Document editor can display and edit the tables. You can select and edit tables through keyboard, mouse, or touch interactions. Document editor exposes a rich set of APIs to perform these operations programmatically.

Create a table

You can create and insert a table at cursor position by specifying the required number of rows and columns.

```
`typescript
```

```
documentedior.editor.insertTable(3,3);
```

```
,
```

The maximum size of row and column is limited to 32767 and 63 respectively.

Insert rows

You can add a row (or several rows) above or below the row at cursor position by using the `insertRow` method. This method accepts the following parameters:

Parameter	Type	Description
-----------	------	-------------

above(optional)	boolean	This is optional and if omitted, it takes the value as false and inserts below the row at cursor position.
-----------------	---------	--

count(optional)	number	This is optional and if omitted, it takes the value as 1.
-----------------	--------	---

Refer to the following sample code.

```
`typescript
```

```
//Inserts a row below the row at cursor position
```

```
documentedior.editor.insertRow();
```

```
//Inserts a row above the row at cursor position
```

```
documentedior.editor.insertRow(false);
```

```
//Inserts three rows below the row at cursor position
```

```
documentedior.editor.insertRow(true, 3)
```

```
,
```

Insert columns

You can add a column (or several columns) to the left or right of the column at cursor position by using the `insertColumn` method. This method accepts the following parameters:

Parameter	Type	Description
-----------	------	-------------

left(optional)	boolean	This is optional and if omitted, it takes the value as false and inserts to the right of column at cursor position.
----------------	---------	---

count(optional)	number	This is optional and if omitted, it takes the value as 1.
-----------------	--------	---

```
`typescript
```

```
//Insert a column to the right of the column at cursor position.
```

```
documentedior.editor.insertColumn();
```

```
//Insert a column to the left of the column at cursor position.
```

```
documentedior.editor.insertColumn(false);
```

```
//Insert two columns to the left of the column at cursor position.
```

```
documentedior.editor.insertColumn(false, 2);
```

```
,
```

Select an entire table

If the cursor position is inside a table, you can select the entire table by using the following sample code.

```
`typescript
```

```
documenteditor.selection.selectTable();
```

```
,
```

Select row

You can select the entire row at cursor position by using the following sample code.

```
`typescript
```

```
documenteditor.selection.selectRow();
```

```
,
```

If current selection spans across cells of different rows, all these rows will be selected.

Select column

You can select the entire column at cursor position by using the following sample code.

```
`typescript
```

```
documenteditor.selection.selectColumn();
```

```
,
```

If current selection spans across cells of different columns, all these columns will be selected.

Select cell

You can select the cell at cursor position by using the following sample code.

```
`typescript
```

```
documenteditor.selection.selectCell();
```

```
,
```

Delete table

Document editor allows to delete the entire table. You can use the `deleteTable()` method of editor instance, if selection is in table.

```
`typescript
```

```
documenteditor.editor.deleteTable();
```

```
,
```

Delete row

Document editor allows to delete the selected number of rows. You can use the `deleteRow()` method of editor instance to delete the selected number of rows, if selection is in table.

```
`typescript
```

```
documenteditor.editor.deleteRow();
```

Delete column

Document editor allows to delete the selected number of columns. You can use the `deleteColumn ()` method of editor instance to delete the selected number of columns, if selection is in table.

`typescript

```
documenteditor.editor.deleteColumn();
```

Merge cells

You can merge cells vertically, horizontally, or combination of both to a single cell. To vertically merge the cells, the columns within selection should be even in left and right directions. To horizontally merge the cells, the rows within selection should be even in top and bottom direction.

`typescript

```
documenteditor.editor.mergeCells()
```

Positioning the table

Document Editor preserves the position properties of the table and displays the table based on position properties. It does not support modifying the position properties. Whereas the table will be automatically moved along with text edited if it is positioned relative to the paragraph.

How to work with tables

The following sample demonstrates how to delete the table row or columns, merge cells and how to bind the API with button.

CSHTML

```
<div id="toolbar"></div>
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableTableDialog(true).EnableSelection(true).EnableSfdtExport(true).Enable
ContextMenu(true).EnableEditorHistory(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.resize();
        updateContainerSize();
        documenteditor.editor.insertTable(2, 2);
        function toolbarButtonClick(arg) {
            switch (arg.item.id) {
                case 'table':
                    //Insert table API to add table
                    documenteditor.editor.insertTable(3, 2);
                    break;
                case 'insert_above':
                    //Insert the specified number of rows to the table above
                    to the row at cursor position
```

```

        documenteditor.editor.insertRow(true, 2);
        break;
    case 'insert_below':
        //Insert the specified number of rows to the table below
        to the row at cursor position
        documenteditor.editor.insertRow();
        break;
    case 'insert_left':
        //Insert the specified number of columns to the table
        left to the column at cursor position
        documenteditor.editor.insertColumn(true, 2);
        break;
    case 'insert_right':
        //Insert the specified number of columns to the table
        right to the column at cursor position
        documenteditor.editor.insertColumn();
        break;
    case 'delete_table':
        //Delete the entire table
        documenteditor.editor.deleteTable();
        break;
    case 'delete_row':
        //Delete the selected number of rows
        documenteditor.editor.deleteRow();
        break;
    case 'delete_column':
        //Delete the selected number of columns
        documenteditor.editor.deleteColumn();
        break;
    case 'merge_cell':
        //Merge the selected cells into one (both vertically and
        horizontally)
        documenteditor.editor.mergeCells();
        break;
    case 'table_dialog':
        //Opens insert table dialog
        documenteditor.showDialog('Table');
        break;
    }
}
function updateContainerSize() {
    document.getElementById('container').style.height =
        window.innerHeight -
document.getElementById('toolbar').offsetHeight + 'px';
}
var toolBar = new ej.navigations.Toolbar({
    clicked: toolbarButtonClick,
    items: [
        {
            prefixIcon: 'e-de-icon-Table',
            tooltipText: 'Insert Table',
            id: 'table',
        },
        { type: 'Separator' },
        {
            prefixIcon: 'e-de-icon-InsertAbove',
            tooltipText: 'Insert new row above',

```

```

        id: 'insert_above',
      },
      {
        prefixIcon: 'e-de-icon-InsertBelow',
        tooltipText: 'Insert new row below',
        id: 'insert_below',
      },
    ],
    { type: 'Separator' },
    {
      prefixIcon: 'e-de-icon-InsertLeft',
      tooltipText: 'Insert new column to the left',
      id: 'insert_left',
    },
    {
      prefixIcon: 'e-de-icon-InsertRight',
      tooltipText: 'Insert new column to the right',
      id: 'insert_right',
    },
    { type: 'Separator' },
    {
      prefixIcon: 'e-de-icon-DeleteTable',
      tooltipText: 'Delete Entire table',
      id: 'delete_table',
    },
    {
      prefixIcon: 'e-de-icon-DeleteRows',
      tooltipText: 'Delete the selected row',
      id: 'delete_row',
    },
    {
      prefixIcon: 'e-de-icon-DeleteColumns',
      tooltipText: 'Delete the selected column',
      id: 'delete_column',
    },
    { type: 'Separator' },
    {
      prefixIcon: 'e-de-icon-Cell',
      tooltipText: 'Merge the selected cells',
      id: 'merge_cell',
    },
    { type: 'Separator' },
    {
      text: 'Dialog',
      tooltipText: 'Open insert table dialog',
      id: 'table_dialog',
    },
  ],
});
toolbar.appendTo('#toolbar');
});
</script>
<style>
  @font-face {
    font-family: 'Sample brower icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSi8AAAEoAAAAVmNtYXDq0OvsAAACfAAAAALBnbHlmA57
AAgAAA6wAAC90aGVhZBF0mKkAAADQAAAAANmhoZWEIUARAAAAAArAAAACRobXR4/AAAAAAAAYAAAD

```

8bG9jYXexHbwAAAMsAAAAgG1heHABZQE/AAABCAAAACBuYW1lChIPawAAMyAAAAAL9cG9zdAI/4kA
AADYgAAADOQABAAAEAAAAAFwEAAAAAAD8wABAAAAAATAAAAPwABAAAAQAAGurlbV8
PPPUACwQAAAAANcxqdwAAAAA1zGp3AAAAAAD8wP0AAAACAACAAAAAATAAAEAAAA/ATMAHAAAAA
AAgAAAAoACgAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnPQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQA
AAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAA
AAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAA
ABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAA
EAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQAQAAAEAAAAAABAAAAAQA
A//8AAAABAAQAQAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAEGATABQAFQAWAB
AGAAZABoAGwAcAB0AHgAfACAAIQAIACMAJAAALACYAJwAoACkAKgArACwALQAUAC8AMAAxADIAMwA
0ADUANgA3ADgAQQA6ADsAPAA9AD4AAAAAFoAngDuAg4CWAJ4ApoCxcgMGA9QD8gVgBcoGSgaMBYo
HYggKCLII3AkICbwJ3An4CjIKvAr4C8QL4AwADEIM6g0MDawNxcg42DoIOpA8yD2YPhA+2EFgQdhE
WEcAR2BI4EyYTXhOUE8AUPhRWFJAUnhVAFegWMBdiF4IXugAOAAAAAPzA7UAwAHAAsADwATABc
AGwAfACMAJwArAC8AMwA3AAALMzUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUhnSEFITU
hJTM1IwUzNSMHMzUjBzM1IwO1Pz+7fX36fX36fX36fX0C7vr6/on6+v6J+voB9AH0/gz+DAF3/ok
C7vr6/si7u/p9fbw/P0t9fX19fX19fX19fX19fX19fX19fX19fX19fX19fX0AAAACAAAAAN2A/MABAA
rAAABEwkBEQmfCTM/BAkBHwYzPwkRIQM4Af7H/sg/AQIDBQYGCakJCQkJCQkIBwEKAQsFBQUGBgY
GDAwFCQgGBgUDAgH9EgO1/JUBZ/6aA2r8lgoJCAgHBwUEAwEBawQFBwEx/s4FBAMDagEBAgIEBQY
ICAgJCgOpAAAABQAAAAAD8wPUAAQACAAnAC4AMgAAJRujJzcHITU3JQ8DHQEfBj8GNS8GDwETESc
HAQcRayERIQ01j51Sdf192gHdAgICAgICBAUGBgYGBgYFBAMBAQMCBwUGCwkFrtp9/sfaPwPo/Bh
9E5xR7c7bQgIDBQYHBgYFBQQDAQEBAQMEBAUGCwoFAgYDAQECaWE9/UDZfQE42gIG/JYDqAAAAAI
AAAAAA/MD8wB/AQUAAAEVDx0rAS8dPQE/HTsBHx0FFR8HAQ8DHwgzPwQBHwc/Hy8fDx4DtQECaWM
FBQUGBwgICQkKCGsLDAwNDQ4ODg4PDw8QEBAQE8QDw8ODg4ODQwNDAsLCgoJCQgIBGcFBQQAeAwI
BAQIDBAQFBQYHCAgJCQoKCwsMDA0NDg4ODg8PDxQEBAQDxAPDw4ODg4NDA0MCwsKCgkJCAGGBwU
FBAQDAgH9UQEEBgCKDA4P/s8GBQIBAgMGCAQFBGsMDAwLBQUFAS0YGBobHB0dHhMTExITERIREBA
QDw8ODg0MDAsLCQkJBwcGBQQDAwEBAQEDAwQFBGcHCQkJCwsMDA0ODg8PEBAQERIRExITExMTExM
SExESERARDw8PDg4NDAwLCwoJCAcHBgUEBAIBAn0QEBAPDw8ODw0ODQ0MDAsLCgoJCQgIBwYFBQU
DAwIBAQIDAUFbQYHCAgJCQoKCwsMDA0NDg0PDg8PDxQEBAQDxAPDw4ODg4NDA0LDAsKCgkJCAC
HBwUFBAQDAgEBAgMEBAUFbwcHCAkJCgoLDAsNDA0ODg4ODw8QDxAQDw8dHRwbGhgY/s4KCgsLCws
KCQUDBAQCAgQEAWUBLRAODAoHBgQBAQEDAwQFBGcHCAoJCwsMDA0ODg8PDxQEERIRExITExMTExM
SExESERAQE8PDg4NDAwLCwkJCQcHBgUEAwMBAQEBAwMEBQYHBwkJCQsLDAwNDg4PDxQEBESEhI
SExMAAAsAAAAA9QD1AADAACwAPABMAFwAbAB8AIwApAC8AACUzNSM1MzUjNzM1IwcZNSMHMzU
jBzM1IwcZNSM3MzUjNTM1IycZIREhESMRFSERIQHhPj4+Pvo+Pn0+Pn0+Pn0+Pn0+Pvo+Pj4++j4
CcfzUPgOo/FjnPj8+Pz4+Pj4+Pj4+Pj8+Pz59/NQDLPzUPgOoAAAAEAAAAAPzA/MAAwAHAAsADwA
ANyE1ITUhNSE1ITUhNSE1IQwD6PwYA+j8GAPO/BgD6PwYDD/6Pvo++j8AAAAAQAQAAADTQO1AAs
AABMJARcJATcJAScJAUSBif53LAGJAYks/ncBiSz+d/53A4n+d/53LAGJ/ncsAYkBiSz+dwGJAAA
FAAAAAAPzA/MAAwAHAA0AEQAVAAA3ITUhJSE1ISUXNyc3JxchNSE1ITUhDAPo/BgBOQKv/VH+x5I
qaWkqpWkv/VH+xwPo/BgMP/o+fZwscHAsHz76PwAABwAAAAAD8wPzAAMABwATABcAGwAfACsAACU
zNSMHMzUjNyMVMxUzNTM1IzUjJSE1ISUzNSMHMzUjFyMVMxUzNTM1IzUjAn0+Pvo/P30+Pj8+Pj/
+DAPo/BgCct4++j8/ft4+Pz4+P8g+Pj4/P/r6Pz59Prw+Pj4+Pz4+P/oAAAAEAAAAAPzA/MAMAA
zAGkApwAAJRUPDi8OPQE/Bx8GAQcnBQ8JFR8OPw81LwkBFQkCJwcXByEBNT8GOWEfBhEzETUvDg8
OA6sBAGMDAwUEBgUGBwYHBwgHBwcHBgYGBQUEBAMCAgEBAgYJChINDRsMCwkIBAL+pOriAsMBNBU
JCgGHBQMBawMFBGcJCQsLDA0NDg4PDw8ODQ0MCwoKCAcGBQQAQMEBgwJCgoVEzT94/7HAVgBloU
wYBX98QECAQIDAUFbGcGBgUFAwMCAT4CagMEBQUGBwCICakJCQkKCQkJCAGHBwYFBQQDAgKuCQk
ICAgHBwCfBQUEAwIBAQEBAgMEBQUGBwCICQkHCQgTFRUFfRQpFRUVExIJAQ3i4iMCSCQSExQ
TExMRERAPDw4ODAsLCQgHBQQDAQEEDBAUHCAkLCwwODg8PEAgRExMTHRMTEiAcQgHUcP67/qgBh6A
odBQBDIOgBfUBAMCAgMEBQUGBv7nArKCKQkCJACIBgYGBAQDAwEBAQEDAwQEAgYGCACJCAkJAAA
AAGAAAAAD8wPzAAMADAAANyE1ISUnBwkBJwCRIwwD6PwYAfTkLAEvAS8s4z8MP+blLP7OATIs5QL
DAAAABgAAAAAD8wPzAB8AXwCfAOIA5QEYAAABFQ8FKwEvBj8GOWEfBQcVHw4/Dy8OIw8OFw8PLw8
/Dx8OJyMPAYcHfW8EJwcfBACXN8DBxc3HwE/Ahc3Jz8DFzcnPwUnBy8DNyChLwM1IycjNSURHw8
hNSEjLwU1ETU/BTMhFTVMz0BLw8hDw4DEgICAwQEBAUFbQQDAwMBAQEBAwMDBAUFbQQEBAwMCAm8
CAGMDBQUGBwYHCAgICQkCJACIBwCwHBgYFBAQDAgEBAQECAwQEAgYGBwCwHCAkICQkCJAGIBwYHBQU
FAwMCAT4BAGMFBQcICQkLCwwMDQ0ODg4MDQwLCgoJBwCwGBQMCQAQECaWUGBwCJCgoLDA0MDg4ODQ0
MDAsLCQkIBwUFAwKiAhQTEhIiKiIJCwoIBDMKNAEDBQYvHDAODg8TFDQUGBQPDwkUNBQSDw0QMBw
vBQUEAQE0CjMICAoQIioiFRESFTgQkP30AQECBAQEAgYGCACICQkJCgGW/moGBgYEBAMCAgMEBQU

GBgGW+j4BAwMEBAYG1gYICAgJCAoJ/mUKCQkJCAcIBgYGBAQEAgEBBgUEBAQDAGICAgMEBAQFBQU
EAwMDAQEDAwMEBQUJCAkIBwcHBgYFBAQDAGEBAGQECawQEBQYGBwCHCAkICQkJCAgHCAYGBgUEBAM
CAgEBAGMEBAUGBgYIBwgICQkODQ0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCwsMDA0NDg4NDQ0MCwo
KCQcHBgQEAgEBAGQEBGcHCQoKCwwNDQ22BAYICikkkQoQERILCTcKGBQTEhsxHA4NCww3FDgDAQE
CATgTOAoLDBEcMBwNERMTDQk4CRQQEBQpJCKLBwYENvqPdzUCGkJCAkHCAYGBgQEAWMBAT8CAwQ
FBQYGAyWGBgUFBAMC+nyCCQkJCQgIBwFVBwUFBAMCAQEBAQIEBAQGBgYIBwgJCQkAAAAABAAAAA
DdgPzAAMABwAiAFMAADchNSEBFQc1AQ8KHQEHnzUvCSM7AR8PBzMVNZUzJz8PMzUjFSE1I4kC7v0
SAbZ+ATIGBgOIBwUFAwMCAf6JAQIBAwQEBQcICGyECgoSEQ4MDAoIBwCFawMDAQEBAm76bQIBAQI
CAwQFBggICgsNDhESFD/9kD8MfQF3UESUATgGBg0NDg4ODg8PDxBfYA8PDw4PDg4NDg0MAwQFBwg
JCgsLDQ4ODhAPIH76jW1+IA8QDg4ODQsLCgkIBwUEA7x9fQACAAAAAAPzA7UAVABgAAABDwUVPwY
7AR8JFQ8QFTM1Iz8SLw8HBQkBFwkBNwkBJwkBA1cODg0MDQwMDAwMDQwNDACNDaoJBAMDAgEBAGQ
GBwkrDDcODAsKcAYCAGH6tAEBAgQECwxAGQ8MBQQEBAICAQEBAgIEBQUHBwgJCgoMDAwNEPylATH
+zzIBJgEmMf7QATax/tr+2gOzAwMFBGcIOQoJBwYEBAlCBAUHBQQGBQcGDgWMCwoKDgorCwwMDQ4
PCAgIJTMHBQYFBQsLMBUPDwgICAKJCgoLDAsLCgkICACGBQQEAWIBAQEm/nH+cCYBgV5/JQGQAY8
m/n4BggAACgAAAAAD8wPzAAMABwALAA8AEwAXABsAHwAJACgAAAEVIZUjFSM1IXUjNQEVIZUjFSM
1IXUjNQEVIZUjFSM1IXUjNQMPAREhA7X6Pvo++gNq+j76PvoDavo++j76PwE5Aq/8GAFF+vr6+vr
6ATj6+vr6+voBOPr6+vr6+vxXA+gAAAAAAQAAAAAD8wPzAIoAABMBNwEhMx8dHQEPHSsBFTM/Hy8
eIyEBJwBjSn+ygIQDw4ODg0ODQWNDawLCwsKCgkJCAgHBwYGBQUDAwMCAQECAwMDBQUGBgCHCAg
JCQoKCwsLDawNDA0ODQ4ODg9eXhIREREREBAQDw8ODg4NDawLCwoKCQgIBwYFBQQDAGEBAGQECawQ
FBQYHCAgJCgoLCwwMDQ4ODg8PEBAQERERERL99wEtKQKY/q0vAQkCAQMDBAQFBgYHBwgICQoJCgs
LDAsMDQ0NDQ4NDg8ODw4ODg0ODQ0MDAwLCwsKCgkJCAgIBGcFBQUDBAICAT8BAQIDBAUFBgICAK
KCgsLDawNDg4ODw8QEBAREREREhIREREREBAQDw8ODg0NDQwLCwoKCQgHBwCFBQMDAwEBCi8AAAU
AAAAAA/MD8wALAA8AEwAXACCAACUjFTMVmZUzNSM1IwEVIzUjFSM1IXUjNQMHESMVIzUjFSM1IXU
jNSMCAH19P3x8PwG1+j76Pvo/A+g/+j76Pvo/yD99fT99AXb6+vr6+vr+yAJx+vr6+vr6AAAcAAA
AAAPUA9QAaWAAHAAADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPafMAVwBbAF8AYwB
nAGsAbwAAJTM1IwcZNSMHMzUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUzNSMFMzUjBTM
1IyUzNSMhMzUjBzM1IwcZNSMFMzUjBzM1IwcZNSMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzU
jNSE1IQOWPj59Pz99Pz+7Pj68Pz99Pz98Pj4Daj4+/ks+Pv5LPj4Daj4+/ks+Pv5LPj4BtT4+AbU
+Pn0/P30/P/6JPz99Pz98Pj4BtT4+AbU+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+A6j8WCw+Pj4+Pj4
+Pj4+Pj4+Pj8/Pz8/Pj8/Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+PgAFAAAAAOWA/M
AAwAfACIAQACFAABByM3JyMVMwCjFTMHfZczBxc3MzUjNZM1IzcnByM3JyUjNScVMxEPBiMhIy8
GET8GMwCRFR8OIT8ONRE1Lw8hDw4CRxJ8EjZwZxJvTA0+DnwMPQ5vZhJvTA0+DnwMPQGIjz76AQI
DBAQGBQf9kAcFBgQEAWIBAQIDBAQGBQdeAgIDBAUFBgCHCAgJCQkKAnAKCQkJCAgHBwYFBQQDAGI
CAgMEBQUg1gCHCAgJCQkJ/mUKCQkJCAgHBwYFBQQDAGIBwn19Pj59P1kJY1kJYj59P1kJY1kJmI8
s+v2vBgYFBQQDAGIDBAUFBgYDLAYGBQUEAwIf/NQKQKICQcIBgYGBAQDAwEBAQEDAwQEBgYGCAC
JCAkJCgJXCQkJCQgIBwFVBgYFBAMCAQEBAQMDBAQGBgYIBwkICQkAAAAADAAAAAAPzA/MACAAMABU
AACUXNxERc3JyUhNSE1JwcXNycHESMBgypTP1Mqnf3tA+j8GAH0UyqcnCpTPvYvTP75AQdML41
9Pq9ML42NL0wBBwAFAAAAAAAPzA/MAAwAHAA0AEQAVAAA3ITUhJSE1ISUXBxc3JwUhNSE1ITUhDAP
o/BgBOQKv/VH+x29vLJubAQ0Cr/1R/scD6PwYDD/6PuxvbyybmX4++j8AAwAAAAADGQ01ACMARgC
bAAABOWEfDg8OKwEREx8PDw8jEQcVESE/GzUvDzU/DzUvECEBzQ0NGRgVFBIDw0LCQgGBQIBAQI
EBGcJCwwODhERExUVF5F7FRQSERAODQwKCQgGBQMCAQECCBAYHCAsLDg4PERITFBZtawEKHx4dDg0
NDQwMDAsLCwoKCQgHBwYGBQQEAWICAQECCBYICQsNDw8RehMUFhYSERAPDg0MCwoIBwYFAwIBAwQ
GBAUFBg0PERMVfHcZGxz+7gHiAgMEBgCHCQsLDQ4PEBITEhEQDw4NDQsKCAgGBAQCAToBdwEBawM
FBQcHCQkLCwwODhASEQ8PDg0LCwoIBwUFAwIBARudP/30AQMGawQFBQYGBwCHCAkJCgoKCgsLDaw
MDQwODQ4WFRQTEhAQDw0MCgoHBgUDAwYHCQkKCw0NDg8PEBAREhILFRUTCQkICRAPDQ0KCQcFAwI
AAAAABAAAAAD8wPzAAMABwALAA8AADchNSE1ITUhNSE1ITUhNSEMAq/9UQPo/BgCr/1RA+j8GAw
/+j76Pvo/AAAAABAAAAA7UD8wADAACACAAANYE1IqERIREDIRehyAJw/ZACr/0SPgNq/Ja9vAI
8/JYDaxXA+gABQAAAAAD8wPzAAMABwATABcAJwAAUjNRMVIZUFIxUzFTM1MzUjNSMFSM1ITM
VIxUzFsvMxUjFSERIqI/+vr6AfN9fT99fT/5+v7H+vr6+vr6AnH9jwFF+voBOPr6Pz59fT59+vr
6+j76Pvo/A+gAAAAACAAAAAN2A/MAAwB4AAA3ITUhExUfHj8eNREjEQcVDxQrAS8UNQMjIQLu/RI
/AQIDAwQFBgYHCAgJCQoKCwsMDA0NDQ4PDg8PDxQEBAQEAA8PDw4PDg0NDQwMCwsKCgkJCAgHBgY
FBAMDAgE+AQICAwMEBQUMDQ8RExMWFgWMDAwNDA0NDA0MDAwMCwsWExMRDw0MCgQDAwICAT4MPwF
3EQ8QDw8PDw4ODg0MDQsMCwoKCQgJBwCGBgUEBAICAQEBAQICBAQFBgYHBwkICQoKCwwLDQwNDg4
ODw8PDxAPEQIy/c4NDQwNDAsMDAsVFBIRDw4LCgQEAgMBAQEBAWIEBAQGCw4PERIUFRcMCwwNDA0
CPwAFAAAAAAAPzA/MAAwAHABMAFwAoAAABFSM1ExUjNQJFTMVmZUzNSM1IyUVIZUDKQE1IzUzNSM
1MzUjNTM1IQK7+fn5/sd9fT98fD8CMvk/ATgBoFr6+vr6+v2PAUT5+QE5+vo/Pn19Pn36+vr8Vz/
6Pvo++j8AAADAAAAAAN2A/MAJQBIAK8AAAEhOWEfBRURFQ8FIyEjLwU1ETU/BTM1FSM1Pw47AR8

2244

1IwcZNSMHMzUjBTM1IywDqPxYA2o+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+AbU+PgG1Pj59Pz99Pz/
+iT8/ft8/fD4+AbU+PgG1Pj7+Sz4+/ks+PgNqPj7+Sz4+/ks+PgNqPj59Pz99Pz/+iT8/ft8/fD4
+AbU+Piw+Pj8/Pz8/Pj8/Pj8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+Pj4+Pj4+Pj4+Pj4
+PgAAAQAAAAAD1APUAAsAAAEhFSErMxEhNSERiWWh/koBtj4Btv5KPGIfPv5KAbY+AbYAAwAAAAA
DdgPzAACAjABIAAABFSE1MxEhESUFbXUzFSE1Mz0BPwg7ARcnDwsjESERiY8ODwIBBgH0Pv2QAVU
GBQQBQIDAX3+in0BAwMEBgUHCQsNEADHBQYKCGwLBwMHAwIB+gLu+gECawUFBggMDgoLCwwMDQw
NDAM4fX39EwLteQMEBQoLBg4NNj8/JxYKCGkIBwCFBAMBNQIDBwCMDgoGEQsNDPyVA2sMDQsMCwo
KDASHBQQEAgEBAGMAAAABgAAAAAD8wPzAAMAQwBHAIcAiWDLAAALITUhBR8PPw8vDw8OASE1KQE
fDz8PLw8PDgEhNSElHw8/Dy8PDw4BRQKv/VH+xwEBAGQEBAYGBggHCAkJCQoKCQkICQcIBgYGBAQ
DAwEBAQEDAwQEBgYGCACJCAkJCgoJCQkIBwGGBgYEBAGQCAQE4Aq/9Uf7HAQECBAMFBgYGBWgICQk
JCgkKCQgJBWgGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQoJCgkJCQgIBWYGBgUDBAIBATgCr/1R/sc
BAQIEAwUGBgYHCAgJCQkKCQoJCAkHCAYGBgQEAWMBAQEBAWMEBAYGBggHCQgJCgkKCQkJCAgHBgY
GBQwEAGFLPh8KCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgoJCQgJBWgGBgYEBAMDAQE
BAQMDBAQGBgYIBwkICQkBTj4KCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgoJCQgJBWg
GBgYEBAMDAQEBAQMDBAQGBgYIBwkICQkBLj8fCgkJCAkHCAYGBgQEAWMBAQEBAWMEBAYGBggHCQg
JCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJAAIAAAAAAPzA/MAAwAHAAAsAEQAVABk
AHQAHAABFSM1IXUjNSMVIzUTMyEVITUBFSM1IXUjNSMVIzUDIREhA7X6Pvo++vo+Ajl8lgNq+j7
6Pvo/A+j8GAFF+vr6+vr6ATj6+gE4+vr6+vr6/FcD6AAABAAAAAAD8wPzAAsADwATABsAAAEbXc
3FzcnNychJwERiXehESMRAYEVITUhESEBg3BwLHBwLHBwLHBwAgb5/on6PwE5AXYBOfwYARlxcCx
wcCxcwSxwCAJw/ooBdv6KAXb+Sz4+AfQAAAAABQAAAAAD1APUAAMABwALAA8AEwAAAREhESMIRE
BESERiXehEQMhESEDlv6JPv6JAyz+iT7+iT4DqPxYAEH+iQF3/okBdwG1/okBd/6JAXf8lgOoAAA
AAIAAAAAA/MDtQBTAf8AAAEpBRU/BjsBHwKVDxAVMzUjPxEvDisBCQIXCQE3CQEnCQEDVw4ODQw
NDAMDAwNDA0MBw0MCgkEAWMCAQECBAYHCREMNw4MCwoIBgICafq0AQECCASMNiMPDAUEBAQCAgE
BAQICBAUFbwcICQoKDAwMDRD8pQEx/s8yASYBJjH+0AEwMf7a/toB/gMDBQYHCDkKCAgGBAQCAgQ
FBwUFBQUHBg4MDAsKCG4KKwsMDA4ODgGICSU0BgYFCwsLKRwODwgICQkJCgoLDAsLCgkICAYGBgQ
EAWIBAZD+cP5xJgGB/n8mAY8Bjyb+fgGCAIAAAAAA/MDtQADAAgAAAEIREDKQERIQJ9/c4/AnE
Bd/wYA3f9EgLu/NQDagAAAAGAAAAA/MD8wADAACACwAPABMAFwAbAB8AACUzNSMFITUhJTM1IwU
hNSElMzUjBSE1ISUzNSMFITUhA7U/P/xAYz81AOpPz/8VwG2/koDqT8//FcCcf2PA6k/P/xAYz
81Aw/Pz/6Pj4++j4+Pvo/Pz8AAQAAAAAC2gPzAAMAACUzASMBJUKBbUgMA+gAABsAAAAA9QD1AA
DAACACwAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMARwBLAE8AUwBXAFsAXwBjAGcAawAAJTM
1IwcZNSMHMzUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUzNSMFMzUjBTM1IzUhNSElMzU
jBTM1IwUzNSMlMzUjBTM1IwUzNSMlMzUjBzM1IwcZNSMHMzUjBzM1IwcZNSMHMzUjA5Y+Pn0/P30
/P7s+Prw/P30/P3w+PgNqPj7+Sz4+/ks+PgNqPj7+Sz4+/ks+PgOo/FgDaj4+/ks+Pv5LPj4Daj4
+/ks+Pv5LPj4Daj4+ft8/ft8/uz4+vD8/ft8/fD4+LD4+Pj4+Pj4+Pj4+Pj4+Pz8/Pz8+Pz8/Pz9
9Pn0/Pz8/Pz4/Pz8/Pz4+Pj4+Pj4+Pj4+Pj4+ABwAAAAA9QD1AADAACACwAPABMAFwAbAB8AIwA
nACsALwAzADcAOwA/AEMARwBLAE8AUwBXAFsAXwBjAGcAawBvAAAlMzUjBzM1IwcZNSMHMzUjBzM
1IwcZNSMlMzUjBTM1IyUzNSMFMzUjJTM1IwcZNSMHMzUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFMzU
jJTM1IwUzNSMlMzUjBzM1IwcZNSMHMzUjBzM1IwcZNSMDMxEjA5Y+Pn0/P30/P7s+Prw/P30/PwL
uPj7+Sz4+AbU+Pv5LPj4BtT4+ft8/ft8/ft8/ft8/ft8/ft8/Au4+Pv5LPj4BtT4+/ks+PgG1Pj5
9Pz99Pz+7Pj68Pz99Pz98Pj4sPj4+Pj4+Pj4+Pj4+Pz8/Pj8/P30+Pj4+Pj4+Pj4+Pj4+ft8/Pz4
/Pz8+Pj4+Pj4+Pj4+Pj78WA0oAAAAAGAAAAA/MD8wAFAAAAEQAZAB0AIwAnADMAADcjFTM1IzM
hNSkBMxUzNTM1IzcjFTM1IzUjMyE1KQEzFTM1IyUhNSErARUzFSMVMzUjNSOJfbw/vAKv/VH+xz8
+P7w/P7w/PvoCr/1R/sd9P7wBOQKv/VH6Pz8/vD8+Sz99Pz8/Prw/Pz4+Pn36Pj4/Pj68AAIAAAA
AA/MC+QCHARQAAAEfBzsBHW0dAg8NKwIvDT0BLwcPBxUfDyE/DzUvDyMPBgUVHw8zPwY9AS8GKwE
vDT0CPw07Ah8ZPwcvEyMPDgK7AQIDBAQFBgddDQwMDAsKCGkIBWYFBQMCAGMFBQYHCAkKCGsMDAw
N+gwNDAsLCgoJCAcGBgQDAgECAwQEBgUHBgYFBQQCAGEBawUGCAkLDA0PDwgREhITAQMUEhIREQ8
PDQwLCQQHBgQCAQMFBgGJCwwNDw8IERISE2chBQYEBAMC/VABawUGCAkLDA0PDwgREhITZwCFBgQ
EAWICAwQEBgUHXQ0NDAsLCgoJCAcGBgQDAgIDBAYGBWgJCgoLCwwNDfkKCQkJCAkICACHBgYGBQU
EBAMCAQIDBAQFBgCGBgUFAwMCAQEDBQYGBWcJCQoKCwwMDA0NDg4O+RMTEHERDw8NDAsJCAyFAwL
bBwUGBAQDAgECAwGGBgICQoKCwsMDQx9DQ0MCwsKCGkIBWYGBAMCAGMEBgYHCAkKCGsLDA0NRQc
FBgQEAWIBAQIDBAQGBQdFFBISEREpdw0MCwkEBwYEAgEDBQYICQsMDQ8PCBESEhOGFBISEREpdw0
MCwkEBwYEAgECAgQFBQaiFRQSEHERDw8NDAsJBACGBAIBAgIEBQUGBgCfBgQEAWIBAgMEBgYHCAk
KCGsLDA0MfQ0MDAwLCgoJCAcGBQUDAgECAgMEBAUFBgYHBWcJCAwMDBMGBQUEAgIBAQICBAUFBgY
TExIRDQwMCwoJCQgHBgUFAwMBAQEDBQYICQsMDQ8PERESEgAAAAQAAAAA/MD8wADAACACwAPAAA
3ITUhJyE1ITChNSEnITUhqAKw/VCCa+j8GJwCsP1QnAPo/BgMP/o++j76PwAFAAAAAAPzA/MAAwA
HAAsAGwAnAAABFSM1IXUjNSMVIzUDMzUzFTM1MxUzNTMVMxEhJSMVMxUzNTM1IzUjA7X6Pvo++j8
/+j76Pvo//BgB9H19Pn19PgI++fn5+fn5/c76+vr6+voCcfO/fX0/fQAAAAASAN4AAQAAAAA

```

AAAAAAAAAAAAAAAAAAAAAAQZAAEEAQAASAAAAGAHABOAAQAACAAAAAAWAZACEAAQAAAAAABAAZADoAAQAAAAA
ABQALAFMAAQAAAAAABgAZAF4AAQAAAAAACGAsAhCAAQAAAAAACwASAKMAAwABBakAAAAACALUAAwA
BBakAAQAYAlCAAwABBakAAgAOAOkaAAwABBakAAwAyAPcAAwABBakABAAYAskAAwABBakABQAWAVs
AAwABBakABgAYAXEAAwABBakACgBYAAmaAAwABBakACwAkAfsgRmluYWwgU2FtcGx1IGJyb3dlciB
pyY29uc1JlZ3VzYXJGaW5hbCBTYWlwbgUGUYnJvd2VyIGljbj25ZRmluYWwgU2FtcGx1IGJyb3dlciB
pyY29uc1Zlcnpb24gMS4wRmluYWwgU2FtcGx1IGJyb3dlciBpyY29uc0ZvbncGZ2VuZXJhdGVkIHV
zaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAARgBpAG4AYQB
sACAAUwBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBSAGUAZWBlAGwAYQByAEY
AaqBuAGEAbAAgAFMAYQBtAHAAbABlACAAyGbyAG8AdwBlAHIAIABPAGMAbwBuAHMARgBpAG4AYQB
sACAAUwBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACA
AMQAuADAARgBpAG4AYQBsACAAUwBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwB
GAG8Abgb0ACAAZwBlAG4AZQByAGEadABlAGQAIABlAHMAAQBuAGcAIABTahKAbgBjAGYAdQBzAGk
AbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwB
uAC4AYwBvAG0AAAAAagAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA/AQIBawEEAQUBBgE
HAQqBCQEKAQsBDAENAQ4BDWEQAREBEgETARQBFQEWARcbGAEZARoBGwEcAR0BHGEfASABIQEIASM
BJAEIASYBJwEOaskBKgErASwBLQEUAS8BMAExAtIBMWe0ATUBNGe3ATgBOQE6ATsBPae9AT4BPwF
AAAtTdHJva2VTdHlsZQhCb29rbWFyaWdqAWN0dXJlBEZpbmQNT3V0c2lkZUJvcmlrcGdkdXN0aWZ
5BUNsb3NldkRlY3JlYXNlSW5kZW50FVBpeGVsQWxpZ25DZW50ZXJUYWJsZS9CYWNrZ3JvdW5kQ29
sb3ILQWxpZ25Cb3R0b20JUGFnZVNldHVwDkhPZ2hsaWdodENvbG9yC1NlcGVyc2NyaXB0BVRhYmx
1BFVuZG8LSW5zZXJ0QmVsb3cJVGV9wQm9yZGVyClBhZ2V0dW1lZXIQWxpZ25DZW50ZXJUYWJsZS9
JbmNyZWZzZUluZGVudARCB2xkcUFsaWduTGVMdAZGb290ZXILSW5zZXJ0UmlnaHQJVW5kZXJsaW5
lCkluc2VydExlZnQETG9jawZIZWFkZXINU3RyaWtldGhyb3VnaAhDbGVhcKfsbAtSaWdodEJvcmlr
lcgpBbGlnblJpZ2h0BE9wZW4KU3Ryb2tlU2l6ZQVQcmludAtEZWxldGVUYWJsZS9QlGlb250Q29sb3I
NSW5zaWRlQm9yZGVycwPEZWXldGVSB3dzCERvd25sb2FkC0xpbnVTcGFjaW5nFELuc2lkZVZlcHR
pY2FsQm9yZGVyCEFSaWduVE9wBFJlZG8MQm90dG9tQm9yZGVyA05ldwVQYXN0ZQdCdWxsZXRxZBEN
lbGwNRGVsZXRLQ29sdWlucwpBbGxCb3JkZXJzCVNlYnNjcmlwdBBTaG93SGlkZVByb3BlcnR5DlR
hYmxlT2ZDb250ZW50Bkl0YWxpYXxzJBnnPzGVib3Jpem9uzGFsYm9yZGVyC0xlZnRCb3JkZXJzCU5
1bWJlcmluZwRMaW5rC0FsaWduQ2VudGVyC0luc2VydEFib3ZlAAAAAA=)
format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="e-de-icon-"],
[class*=" e-de-icon-"] {
font-family: 'Sample browser icons' !important;
}
.e-de-icon-Table:before {
content: "\e70e";
}
.e-de-icon-InsertBelow:before {
content: "\e710";
}
.e-de-icon-InsertRight:before {
content: "\e718";
}
.e-de-icon-InsertLeft:before {
content: "\e71a";
}
.e-de-icon-DeleteTable:before {
content: "\e724";
}
.e-de-icon-DeleteRows:before {
content: "\e727";
}
.e-de-icon-Cell:before {
content: "\e731";
}

```

```
.e-de-icon-DeleteColumns:before {
    content: "\e732";
}
.e-de-icon-InsertAbove:before {
    content: "\e73d";
}
</style>
```

TABLE.CS

See Also

- [Feature modules](#)
- [Insert table dialog](#)

Table of contents

The table of contents in a document is same as the list of chapters at the beginning of a book. It lists each heading in the document and the page number, where that heading starts with various options to customize the appearance.

Inserting table of contents

Document editor exposes an API to insert table of contents at cursor position programmatically. You can specify the settings for table of contents explicitly. Otherwise, the default settings will be applied.

TableOfContentsSettings contain the following properties:

- **startLevel**: Specifies the start level for constructing table of contents.
- **endLevel**: Specifies the end level for constructing table of contents.
- **includeHyperlink**: Specifies whether the link for headings is included.
- **includePageNumber**: Specifies whether the page number of the headings is included.
- **rightAlign**: Specifies whether the page number is right aligned.
- **tabLeader**: Specifies the tab leader styles such as none, dot, hyphen, and underscore.
- **includeOutlineLevels**: Specifies whether the outline levels are included.

```
`javascript
var tocSettings=
{
startLevel: 1, endLevel: 3, includeHyperlink: true, includePageNumber: true, rightAlign: true
};
documenteditor.editor.insertTableOfContents(tocSettings);
`
```

CSHTML

```

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var documentString =
'{"sections":[{"blocks":[{"paragraphFormat":{"styleName":"Heading
1"},"inlines":[{"text":"Headin"}, {"name":"_GoBack", "bookmarkType":0}, {"name"
: "_GoBack", "bookmarkType":1}, {"text":"g1"}]}, {"paragraphFormat":{"styleName"
:"Heading
2"},"inlines":[{"text":"Heading2"}]}, {"paragraphFormat":{"styleName":"Headin
g
3"},"inlines":[{"text":"Heading3"}]}, {"paragraphFormat":{"styleName":"Headin
g
4"},"inlines":[{"text":"Heading4"}]}, {"paragraphFormat":{"styleName":"Headin
g
5"},"inlines":[{"text":"Heading5"}]}, {"paragraphFormat":{"styleName":"Headin
g
6"},"inlines":[{"text":"Heading6"}]}, {"paragraphFormat":{"styleName":"Normal
"},"inlines":[{"text":"Normal"}]}], "headersFooters": {}, "sectionFormat": {"hea
derDistance":36.0, "footerDistance":36.0, "pageWidth":612.0, "pageHeight":792.0
, "leftMargin":72.0, "rightMargin":72.0, "topMargin":72.0, "bottomMargin":72.0, "
differentFirstPage":false, "differentOddAndEvenPages":false}}, "characterForm
at":{"fontSize":11.0, "fontFamily":"Calibri"}, "paragraphFormat":{"afterSpacin
g":8.0, "lineSpacing":1.0791666507720947, "lineSpacingType":"Multiple"}, "backg
round":{"color":"#FFFFFFF"}, "styles":[{"type":"Paragraph", "name":"Normal", "
next":"Normal"}, {"type":"Paragraph", "name":"Heading
1", "basedOn":"Normal", "next":"Normal", "link":"Heading 1
Char", "characterFormat":{"fontSize":16.0, "fontFamily":"Calibri
Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":12.0, "aft
erSpacing":0.0, "outlineLevel":"Level1"}}, {"type":"Paragraph", "name":"Heading
2", "basedOn":"Normal", "next":"Normal", "link":"Heading 2
Char", "characterFormat":{"fontSize":13.0, "fontFamily":"Calibri
Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afte
rSpacing":0.0, "outlineLevel":"Level2"}}, {"type":"Paragraph", "name":"Heading
3", "basedOn":"Normal", "next":"Normal", "link":"Heading 3
Char", "characterFormat":{"fontSize":12.0, "fontFamily":"Calibri
Light", "fontColor":"#1F3763FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afte
rSpacing":0.0, "outlineLevel":"Level3"}}, {"type":"Paragraph", "name":"Heading
4", "basedOn":"Normal", "next":"Normal", "link":"Heading 4
Char", "characterFormat":{"italic":true, "fontFamily":"Calibri
Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afte
rSpacing":0.0, "outlineLevel":"Level4"}}, {"type":"Paragraph", "name":"Heading
5", "basedOn":"Normal", "next":"Normal", "link":"Heading 5
Char", "characterFormat":{"fontFamily":"Calibri
Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afte
rSpacing":0.0, "outlineLevel":"Level5"}}, {"type":"Paragraph", "name":"Heading
6", "basedOn":"Normal", "next":"Normal", "link":"Heading 6
Char", "characterFormat":{"fontFamily":"Calibri
Light", "fontColor":"#1F3763FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afte
rSpacing":0.0, "outlineLevel":"Level6"}}, {"type":"Character", "name":"Default
Paragraph Font"}, {"type":"Character", "name":"Heading 1
Char", "basedOn":"Default Paragraph
Font", "characterFormat":{"fontSize":16.0, "fontFamily":"Calibri
Light", "fontColor":"#2F5496FF"}}, {"type":"Character", "name":"Heading 2

```

```

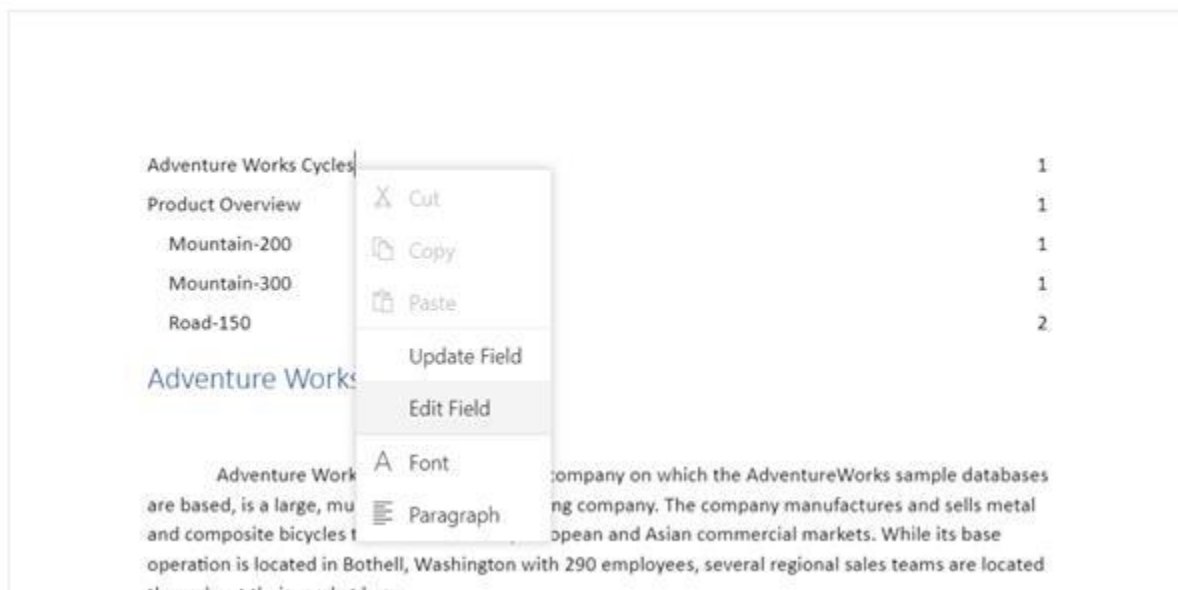
Char", "basedOn": "Default Paragraph
Font", "characterFormat": { "fontSize": 13.0, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF" } }, { "type": "Character", "name": "Heading 3
Char", "basedOn": "Default Paragraph
Font", "characterFormat": { "fontSize": 12.0, "fontFamily": "Calibri
Light", "fontColor": "#1F3763FF" } }, { "type": "Character", "name": "Heading 4
Char", "basedOn": "Default Paragraph
Font", "characterFormat": { "italic": true, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF" } }, { "type": "Character", "name": "Heading 5
Char", "basedOn": "Default Paragraph
Font", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF" } }, { "type": "Character", "name": "Heading 6
Char", "basedOn": "Default Paragraph
Font", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#1F3763FF" } } } ]';
    editor.open(documentString);
    // To insert table of content in document editor
    var tocSettings =
    {
        startLevel: 1, endLevel: 3, includeHyperlink: true,
includePageNumber: true, rightAlign: true
    };
    documenteditor.editor.insertTableOfContents(tocSettings);
});
</script>

```

TABLE-OF-CONTENTS.CS

Update or edit table of contents

You can update or edit the table of contents using the built-in context menu shown up by right-clicking it.



- **Update Field:** Updates the headings in table of contents with same settings by searching the entire document.
- **Edit Field:** Opens the built-in table of contents dialog and allows to modify its settings.

You can also do it programmatically by using the exposed API.

```
`typescript
documenteditor.open(""); /Open any existing document/
var tocSettings =
{
startLevel: 1, endLevel: 3, includeHyperlink: true, includePageNumber: true, rightAlign: true
};
documenteditor.editorModule.insertTableOfContents(tocSettings);
`
```

Note: Same method is used for inserting, updating, and editing table of contents. This will work based on the current element at cursor position and the optional settings parameter. If table of contents is present at cursor position, the update operation will be done based on the optional settings parameter. Otherwise, the insert operation will be done.

See Also

- [Table of contents dialog](#)

Headers and Footers

Document editor supports headers and footers in its document. Each section in the document can have the following types of headers and footers:

- **First page:** Used only on the first page of the section.
- **Even pages:** Used on all even numbered pages in the section.
- **Default:** Used on all pages of the section, where the first or even pages are not applicable or not specified.

You can define this by setting format properties of the corresponding section using the following sample code.

```
`typescript
//Defines whether different header footer is required for first page of the section
documenteditor.selection.sectionFormat.differentFirstPage= true;
//Defines whether different header footer is required for odd and even pages in the section
documenteditor.selection.sectionFormat.differentOddAndEvenPages= true;
`
```

Go to header footer region

Double click in header or footer region to move the selection into it.


```
`typescript
documenteditor.selection.goToHeader();
`
```

```
`typescript
documenteditor.selection.goToFooter();
`
```

[Link to previous](#)

Link to previous is enabled by default when document has more than one section. If you're using different headers and footers such as different first page or different odd and even pages, they can't be linked together because they're all separate.

Before setting or getting the link to previous value, use the ['goToHeader'](#) or ['goToFooter'](#) API to move the current selection to the header or footer region.

You can get or set the default header footer link to previous value of a section at cursor position by using the following sample code.

```
`typescript
container.documentEditor.selection.sectionFormat.oddPageHeader.linkToPrevious = false;
container.documentEditor.selection.sectionFormat.oddPageFooter.linkToPrevious = false;
`
```

In case the document has different header and footer types, such as different first page, odd, and even pages.

```
`typescript
// Different first page
container.documentEditor.selection.sectionFormat.firstPageHeader.linkToPrevious = false;
container.documentEditor.selection.sectionFormat.firstPageFooter.linkToPrevious = false;
//Even page
container.documentEditor.selection.sectionFormat.firstPageHeader.linkToPrevious = false;
container.documentEditor.selection.sectionFormat.firstPageFooter.linkToPrevious = false;
`
```

Note: When there is more than one section in the document, the Link to Previous option becomes available. By default, this feature is disabled state in UI and set to return false for the first section.

[Header and footer distance](#)

You can define the distance of header region content from the top of the page.

```
`typescript
documenteditor.selection.sectionFormat.headerDistance= 36;
`
```

Same way, you can define the distance of footer region content from the bottom of the page.

```
`typescript
documenteditor.selection.sectionFormat.footerDistace=36;
`
```

Close header footer region

Move the selection to the document body from header or footer region by double clicking or tapping the document area.

```
`typescript
documenteditor.selection.closeHeaderFooter()
`
```

See Also

- [Working with Section Formatting](#)

Working with Text Formatting

Document editor supports several formatting options for text like bold, italic, font color, highlight color, and more. This section describes how to modify the formatting for selected text in detail.

Bold

The bold formatting for selected text can be get or set by using the following sample code.

```
`typescript
//Gets the value for bold formatting of selected text.
let bold : boolean = documenteditor.selection.characterFormat.bold;
//Sets bold formatting for selected text.
documenteditor.selection.characterFormat.bold = true;
`
```

You can toggle the bold formatting based on existing value at selection.

```
`typescript
documenteditor.editor.toggleBold();
`
```

Italic

The Italic formatting for selected text can be get or set by using the following sample code.

```
`typescript
documenteditor.selection.characterFormat.italic= true|false;
`
```

You can toggle the Italic formatting based on existing value at selection.

```
`typescript
```

```
documenteditor.editor.toggleItalic();
```

,

Underline property

The underline style for selected text can be get or set by using the following sample code.

```
`typescript
```

```
documenteditor.selection.characterFormat.underline='Single' | 'None';
```

,

You can toggle the underline style of selected text based on existing value at selection by specifying a value.

```
`typescript
```

```
documenteditor.editor.toggleUnderline('Single');
```

,

Strikethrough property

The strikethrough style for selected text can be get or set by using the following sample code.

```
`typescript
```

```
documenteditor.selection.characterFormat.strikethrough='Single' | 'Normal';
```

,

You can toggle the strikethrough style of selected text based on existing value at selection by specifying a value.

```
`typescript
```

```
documenteditor.editor.toggleStrikethrough();
```

,

Superscript property

The selected text can be made superscript by using the following sample code.

```
`typescript
```

```
documenteditor.selection.characterFormat.baselineAlignment='Superscript';
```

,

Toggle the selected text as superscript or normal using the following sample code.

```
`typescript
```

```
documenteditor.editor.toggleSuperscript();
```

,

Subscript property

The selected text can be made subscript by using the following sample code.

```
`typescript
```

```
documenteditor.selection.characterFormat.baselineAlignment='Subscript';
```

`

Toggle the selected text as subscript or normal using the following sample code.

```
`typescript
documenteditor.editor.toggleSubscript();
```

`

You can make a subscript or superscript text as normal using the following code.

```
`typescript
documenteditor.selection.characterFormat.baselineAlignment='Normal';
```

`

Size

The size of selected text can be get or set using the following code.

```
`typescript
documenteditor.selection.characterFormat.fontSize= 32;
```

`

Color

The color of selected text can be get or set using the following code.

```
`typescript
documenteditor.selection.characterFormat.fontColor= 'Pink';
documenteditor.selection.characterFormat.fontColor= '#FFC0CB';
```

`

Font

The font style of selected text can be get or set using the following sample code.

```
`typescript
documenteditor.selection.characterFormat.fontFamily= 'Arial';
```

`

Highlight color

The highlight color of the selected text can be get or set using the following sample code.

```
`typescript
documenteditor.selection.characterFormat.highlightColor= 'Pink';
documenteditor.selection.characterFormat.highlightColor= '#FFC0CB';
```

`

Toolbar with options for text formatting

CSHTML

```
<div id="toolbar"></div>
<div id="documenteditor" style="width:100%;height:100%">
```

```

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).EnableContextMenu(true).Enable
EditorHistory(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.resize();
        updateContainerSize();
        function toolbarButtonClick(arg) {
            switch (arg.item.id) {
                case 'bold':
                    //Toggles the bold of selected content
                    documenteditor.editor.toggleBold();
                    break;
                case 'italic':
                    //Toggles the Italic of selected content
                    documenteditor.editor.toggleItalic();
                    break;
                case 'underline':
                    //Toggles the underline of selected content
                    documenteditor.editor.toggleUnderline('Single');
                    break;
                case 'strikethrough':
                    //Toggles the strikethrough of selected content
                    documenteditor.editor.toggleStrikethrough();
                    break;
                case 'subscript':
                    //Toggles the subscript of selected content
                    documenteditor.editor.toggleSubscript();
                    break;
                case 'superscript':
                    //Toggles the superscript of selected content
                    documenteditor.editor.toggleSuperscript();
                    break;
            }
        }
        function updateContainerSize() {
            document.getElementById('container').style.height =
                window.innerHeight -
document.getElementById('toolbar').offsetHeight + 'px';
        }
        //To change the font Style of selected content
        function changeFontFamily(args) {
            documenteditor.selection.characterFormat.fontFamily =
args.value;
            documenteditor.focusIn();
        }
        //To Change the font Size of selected content
        function changeFontSize(args) {
            documenteditor.selection.characterFormat.fontSize = args.value;
            documenteditor.focusIn();
        }
        //To Change the font Color of selected content

```

```

function changeFontColor(args) {
    documenteditor.selection.characterFormat.fontColor =
args.currentValue.hex;
    documenteditor.focusIn();
}
documenteditor.selectionChange = function () {
    setTimeout(function () { onSelectionChange(); }, 20);
};
//Selection change to retrieve formatting
function onSelectionChange() {
    if (documenteditor.selection) {
        enableDisableFontOptions();
        // #endregion
    }
}
function enableDisableFontOptions() {
    var characterformat = documenteditor.selection.characterFormat;
    var properties = [characterformat.bold, characterformat.italic,
characterformat.underline, characterformat.strikeThrough];
    var toggleBtnId = ["bold", "italic", "underline",
"strikethrough"];
    for (var i = 0; i < properties.length; i++) {
        changeActiveState(properties[i], toggleBtnId[i]);
    }
}
function changeActiveState(property, btnId) {
    var toggleBtn = document.getElementById(btnId);
    if ((typeof (property) == 'boolean' && property == true) ||
(typeof (property) == 'string' && property != 'None'))
        toggleBtn.classList.add("e-btn-toggle");
    else {
        if (toggleBtn.classList.contains("e-btn-toggle"))
            toggleBtn.classList.remove("e-btn-toggle");
    }
}
var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria
Math', 'Candara', 'Courier New', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18',
'20', '22', '24', '26', '28', '36', '48', '72', '96'];
var toolBar = new ej.navigations.Toolbar({
    clicked: toolbarButtonClick,
    items: [
        {
            prefixIcon: 'e-de-icon-Bold',
            tooltipText: 'Bold',
            id: 'bold',
        },
        {
            prefixIcon: 'e-de-icon-Italic',
            tooltipText: 'Italic',
            id: 'italic',
        },
        {
            prefixIcon: 'e-de-icon-Underline',
            tooltipText: 'Underline',

```

```

        id: 'underline',
      },
      {
        prefixIcon: 'e-de-icon-Strikethrough',
        tooltipText: 'Strikethrough',
        id: 'strikethrough',
      },
      {
        prefixIcon: 'e-de-icon-Subscript',
        tooltipText: 'Subscript',
        id: 'subscript',
      },
      {
        prefixIcon: 'e-de-icon-Superscript',
        tooltipText: 'Superscript',
        id: 'superscript',
      },
    ],
    { type: 'Seperator' },
    {
      type: 'Input',
      template: new ColorPicker({
        value: '#000000',
        showButtons: true,
        change: changeFontColor
      }),
    },
    { type: 'Seperator' },
    {
      type: 'Input',
      template: new ComboBox({
        dataSource: fontStyle,
        width: 120,
        index: 2,
        allowCustom: true,
        change: changeFontFamily,
        showClearButton: false,
      }),
    },
    {
      type: 'Input',
      template: new ComboBox({
        dataSource: fontSize,
        width: 80,
        allowCustom: true,
        index: 2,
        change: changeFontSize,
        showClearButton: false,
      }),
    },
  ],
});
toolBar.appendTo('#toolbar');
});
</script>
<style>
  #container {
    width: 100%;

```

```
        height: 100%
    }
</style>
```

TEXT-FORMAT.CS

See Also

- [Feature modules](#)
- [Font dialog](#)
- [Keyboard shortcuts](#)

Working with Paragraph Formatting

Document editor supports various paragraph formatting options such as text alignment, indentation, paragraph spacing, and more.

Indentation

You can modify the left or right indentation of selected paragraphs using the following sample code.

```
`typescript
documenteditor.selection.paragraphFormat.leftIndent= 24;
documenteditor.selection.paragraphFormat.rightIndent= 24;
`
```

Special indentation

You can define special indent for the first line of the paragraph using the following sample code.

```
`typescript
documenteditor.selection.paragraphFormat.firstLineIndent= 24;
`
```

Increase indent

You can increase the left indent of selected paragraphs by a factor of 36 points using the following sample code.

```
`typescript
documenteditor.editor.increaseIndent()
`
```

Decrease indent

You can decrease the left indent of selected paragraphs by a factor of 36 points using the following sample code.

```
`typescript
documenteditor.editor.decreaseIndent()
```


`

Text alignment

You can get or set the text alignment of selected paragraphs using the following sample code.

`typescript

```
documenteditor.selection.paragraphFormat.textAlignment= 'Center' | 'Left' | 'Right' | 'Justify';
```

`

You can toggle the text alignment of selected paragraphs by specifying a value using the following sample code.

`typescript

```
documenteditor.editor.toggleTextAlignment('Center' | 'Left' | 'Right' | 'Justify');
```

`

Line spacing and its type

You can define the line spacing and its type for selected paragraphs using the following sample code.

`typescript

```
documenteditor.selection.paragraphFormat.lineSpacingType='AtLeast';
```

```
documenteditor.selection.paragraphFormat.lineSpacing= 6;
```

`

Paragraph spacing

You can define the spacing before or after the paragraph by using the following sample code.

`typescript

```
documenteditor.selection.paragraphFormat.beforeSpacing= 24;
```

```
documenteditor.selection.paragraphFormat.afterSpacing= 24;
```

`

You can also set automatic spacing before and after the paragraph by using the following sample code.

`typescript

```
documenteditor.selection.paragraphFormat.spaceBeforeAuto = true;
```

```
documenteditor.selection.paragraphFormat.spaceAfterAuto = true;
```

`

Note: If auto spacing property is enabled, then value defined in the `beforeSpacing` and `afterSpacing` property will not be considered.

Pagination properties

You can enable or disable the following pagination properties for the paragraphs in a Word document.

- Widow/Orphan control - whether the first and last lines of the paragraph are to remain on the same page as the rest of the paragraph when paginating the document.

- Keep with next - whether the specified paragraph remains on the same page as the paragraph that follows it while paginating the document.
- Keep lines together - whether all lines in the specified paragraphs remain on the same page while paginating the document.

```
`typescript
```

```
documenteditor.selection.paragraphFormat.widowControl = false;  
documenteditor.selection.paragraphFormat.keepWithNext = true;  
documenteditor.selection.paragraphFormat.keepLinesTogether = true;  
`
```

Paragraph Border

You can apply borders to the paragraphs in a Word document. Using borders, decorate the paragraphs to set them apart from other paragraphs in the document.

The following example code illustrates how to apply box border for the selected paragraphs.

```
`typescript
```

```
// left
```

```
documenteditor.selection.paragraphFormat.borders.left.lineStyle = 'Single';  
documenteditor.selection.paragraphFormat.borders.left.lineWidth = 3;  
documenteditor.selection.paragraphFormat.borders.left.color = "#000000";
```

```
//right
```

```
documenteditor.selection.paragraphFormat.borders.right.lineStyle = 'Single';  
documenteditor.selection.paragraphFormat.borders.right.lineWidth = 3;  
documenteditor.selection.paragraphFormat.borders.right.color = "#000000";
```

```
//top
```

```
documenteditor.selection.paragraphFormat.borders.top.lineStyle = 'Single';  
documenteditor.selection.paragraphFormat.borders.top.lineWidth = 3;  
documenteditor.selection.paragraphFormat.borders.top.color = "#000000";
```

```
//bottom
```

```
documenteditor.selection.paragraphFormat.borders.bottom.lineStyle = 'Single';  
documenteditor.selection.paragraphFormat.borders.bottom.lineWidth = 3;  
documenteditor.selection.paragraphFormat.borders.bottom.color = "#000000";  
`
```

Note: At present, the Document editor component displays all the border styles as single line. But you can apply any border style and get the proper display in Microsoft Word app when opening the exported Word document.

Show or Hide Paragraph marks

You can show or hide the hidden formatting symbols like spaces, tab, paragraph marks, and breaks in Document editor component. These marks help identify the start and end of a paragraph and all the hidden formatting symbols in a Word document.

The following example code illustrates how to show or hide paragraph marks.

```
`typescript
```

```
documenteditor.documentEditorSettings.showHiddenMarks = true;
```

Toolbar with paragraph formatting options

CSHTML

```
<div id="toolbar"></div>
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).EnableContextMenu(true).Enable
EditorHistory(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.resize();
        updateContainerSize();
        function toolbarButtonClick(arg) {
            switch (arg.item.id) {
                case 'AlignLeft':
                    //Toggle the Left alignment for selected or current
paragraph
                    documenteditor.editor.toggleTextAlignment('Left');
                    break;
                case 'AlignRight':
                    //Toggle the Right alignment for selected or current
paragraph
                    documenteditor.editor.toggleTextAlignment('Right');
                    break;
                case 'AlignCenter':
                    //Toggle the Center alignment for selected or current
paragraph
                    documenteditor.editor.toggleTextAlignment('Center');
                    break;
                case 'Justify':
                    //Toggle the Justify alignment for selected or current
paragraph
                    documenteditor.editor.toggleTextAlignment('Justify');
                    break;
                case 'IncreaseIndent':
                    //Increase the left indent of selected or current
paragraph
                    documenteditor.editor.increaseIndent();
                    break;
```

```

        case 'DecreaseIndent':
            //Decrease the left indent of selected or current
paragraph
            documenteditor.editor.decreaseIndent();
            break;
        case 'ClearFormat':
            documenteditor.editor.clearFormatting();
            break;
        case 'ShowParagraphMark':
            //Show or hide the hidden characters like spaces, tab,
paragraph marks, and breaks.
            documenteditor.documentEditorSettings.showHiddenMarks =
!documenteditor.documentEditorSettings.showHiddenMarks;
            break;
    }
}
//Change the line spacing of selected or current paragraph
function lineSpacingAction(args) {
    var text = args.item.text;
    switch (text) {
        case 'Single':
            documenteditor.selection.paragraphFormat.lineSpacing =
1;

            break;
        case '1.15':
            documenteditor.selection.paragraphFormat.lineSpacing =
1.15;

            break;
        case '1.5':
            documenteditor.selection.paragraphFormat.lineSpacing =
1.5;

            break;
        case 'Double':
            documenteditor.selection.paragraphFormat.lineSpacing =
2;

            break;
    }
    setTimeout(function () {
        documenteditor.focusIn();
    }, 30);
}
documenteditor.selectionChange = function () {
    setTimeout(function () {
        onSelectionChange();
    }, 20);
};
// Selection change to retrieve formatting
function onSelectionChange() {
    if (documenteditor.selection) {
        var paragraphFormat =
documenteditor.selection.paragraphFormat;
        var toggleBtnId = ['AlignLeft', 'AlignCenter', 'AlignRight',
'Justify', 'ShowParagraphMark'];
        for (var i = 0; i < toggleBtnId.length; i++) {
            var toggleBtn = document.getElementById(
toggleBtnId[i]
            );

```

```

        toggleBtn.classList.remove('e-btn-toggle');
    }
    if (paragraphFormat.textAlignment === 'Left') {
        document.getElementById('AlignLeft').classList.add('e-
btn-toggle');
    } else if (paragraphFormat.textAlignment === 'Right') {
        document.getElementById('AlignRight').classList.add('e-
btn-toggle');
    } else if (paragraphFormat.textAlignment === 'Center') {
        document
            .getElementById('AlignCenter')
            .classList.add('e-btn-toggle');
    } else {
        document.getElementById('Justify').classList.add('e-btn-
toggle');
    }
    if(documenteditor.documentEditorSettings.showHiddenMarks) {
document.getElementById('ShowParagraphMark').classList.add('e-btn-toggle');
    }
    // #endregion
}
}
//Toolbar configuration to add paragraph formatting options
var toolBar = new ej.navigations.Toolbar({
    clicked: toolbarButtonClick,
    items: [
        {
            prefixIcon: 'e-de-icon-AlignLeft',
            tooltipText: 'Align Left',
            id: 'AlignLeft',
        },
        {
            prefixIcon: 'e-de-icon-AlignCenter',
            tooltipText: 'Align Center',
            id: 'AlignCenter',
        },
        {
            prefixIcon: 'e-de-icon-AlignRight',
            tooltipText: 'Align Right',
            id: 'AlignRight',
        },
        {
            prefixIcon: 'e-de-icon-Justify',
            tooltipText: 'Justify',
            id: 'Justify',
        },
        {
            prefixIcon: 'e-de-icon-IncreaseIndent',
            tooltipText: 'Increase Indent',
            id: 'IncreaseIndent',
        },
        {
            prefixIcon: 'e-de-icon-DecreaseIndent',
            tooltipText: 'Decrease Indent',
            id: 'DecreaseIndent',
        },
    ],
});

```

```

        { type: 'Seperator' },
        {
            id: 'lineSpacing',
        }, {
            prefixIcon: 'e-de-icon-ClearAll',
            tooltipText: 'ClearFormatting',
            id: 'ClearFormat',
        },
        { type: 'Seperator' },
        {
            prefixIcon: 'e-de-e-paragraph-mark e-icons',
            tooltipText: 'Show the hidden characters like spaces,
tab, paragraph marks, and breaks.(Ctrl + *)',
            id: 'ShowParagraphMark',
        }
    ],
});
toolBar.appendTo('#toolbar');
var items = [
    {
        text: 'Single',
    },
    {
        text: '1.15',
    },
    {
        text: '1.5',
    },
    {
        text: 'Double',
    },
];
var dropdown = new ej.splitbuttons.DropDownButton({
    items: items,
    iconCss: 'e-de-icon-LineSpacing',
    select: lineSpacingAction,
});
dropdown.appendTo('#lineSpacing');
});
</script>
<style>
    #container {
        width: 100%;
        height: 100%
    }
</style>

```

PARAGRAPH-FORMAT.CS

See Also

- [Feature modules](#)

- [Paragraph dialog](#)
- [Keyboard shortcuts](#)

Styles

Styles are useful for applying a set of formatting consistently throughout the document. In document editor, styles are created and added to a document programmatically or via the built-in Styles dialog.

Styles definition overview

A Style in document editor should have the following properties:

- **name:** Name of the style. All styles in a document have a unique name, which is used as an identifier when applying the style.
- **type:** Specifies the document elements that the style will target. For example, paragraph or character.
- **next:** Specifies that the current style inherits the style set to this property. This is how hierarchical styles are defined.
- **link:** Provides a relation between the paragraph and character style.
- **characterFormat:** Specifies the properties of paragraph and character style.
- **paragraphFormat:** Specifies the properties of paragraph style.
- **basedOn:** Specifies that the current style inherits the style set to this property. This is how hierarchical styles are defined. It can be optional.

Note: The style type should match the inherited style type. For example, it is not possible to have a character style inherit a paragraph style.

Default style

The default style for span and paragraph properties is normal. It internally inherits the default style of the document loaded or document editor component.

Style hierarchy

Each style initially checks its local value for the property that is being evaluated and turns to the style it is based on. If no local value is found, it turns to its default style.

Style inheritance of different styles are listed as follows:

Character style

Character styles are based only on other character styles.

The inheritance is: Character properties are inherited from the base character style.

Paragraph style

Paragraph styles are based on other paragraph styles or on linked styles. When a paragraph style is based on another paragraph style, the inheritance of the properties is as follows:

- Paragraph properties are inherited from the base paragraph style.
- Span properties are inherited from the base paragraph style.

When a paragraph style is based on a linked style, the inheritance of the properties is as follows:

- Paragraph properties are inherited from the paragraph style part in its base linked style.
- Span properties are inherited from the span style part in its base linked style.

Linked style

Linked styles are composite styles and their components are paragraph and character styles with link between them. To apply paragraph properties, take the properties from the linked paragraph style. Similarly, to apply character properties, take the properties from linked character style. Linked styles are based on other linked styles or on paragraph styles.

When a linked style is based on a paragraph style, the hierarchy of the properties is as follows:

- Paragraph properties are inherited from the 'basedOn' paragraph style.
- Character properties are inherited from the 'basedOn' paragraph style.

When a linked style is based on another linked style, the hierarchy of the properties is as follows:

- Paragraph properties are inherited from the paragraph style part in its base linked style.
- Span properties are inherited from the span style part in its base linked style.

Defining new styles

New Styles are defined and added to the style collection of the document. In this way, they will be discovered by the default UI and applied to the parts of a document.

Defining a character style

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var styleJson = {
            "type": "Character",
            "name": "New CharacterStyle",
            "basedOn": "Default Paragraph Font",
            "characterFormat": {
                "fontSize": 16.0,
                "fontFamily": "Calibri Light",
                "fontColor": "#2F5496",
                "bold": true,
                "italic": true,
                "underline": "Single"
            }
        };
        documentEditor.editor.createStyle(JSON.stringify(styleJson));
    });
</script>
```

CHARACTER-STYLE.CS

*Defining a paragraph style***CSHTML**

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
        document.getElementById("container").ej2_instances[0];
        var styleJson = {
            "type": "Paragraph",
            "name": "New ParagraphStyle",
            "basedOn": "Normal",
            "characterFormat": {
                "fontSize": 16.0,
                "fontFamily": "Calibri Light",
                "fontColor": "#2F5496",
                "bold": true,
                "italic": true,
                "underline": "Single"
            },
            "paragraphFormat": {
                "leftIndent": 0.0,
                "rightIndent": 0.0,
                "firstLineIndent": 0.0,
                "beforeSpacing": 12.0,
                "afterSpacing": 0.0,
                "lineSpacing": 1.0791666507720947,
                "lineSpacingType": "Multiple",
                "textAlignment": "Left",
                "outlineLevel": "Level1"
            }
        };
        documenteditor.editor.createStyle(JSON.stringify(styleJson));
    });
</script>
```

PARAGRAPH-STYLE.CS*Defining a linked style***CSHTML**

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
        document.getElementById("container").ej2_instances[0];
        var styleJson = {
            "type": "Paragraph",
            "name": "New Linked",
```

```

        "basedOn": "Normal",
        "next": "Normal",
        "link": "New Linked Char",
        "characterFormat": {
            "fontSize": 16.0,
            "fontFamily": "Calibri Light",
            "fontColor": "#2F5496"
        },
        "paragraphFormat": {
            "leftIndent": 0.0,
            "rightIndent": 0.0,
            "firstLineIndent": 0.0,
            "beforeSpacing": 12.0,
            "afterSpacing": 0.0,
            "lineSpacing": 1.0791666507720947,
            "lineSpacingType": "Multiple",
            "textAlignment": "Left",
            "outlineLevel": "Level1"
        }
    };
    documentEditor.editor.createStyle(JSON.stringify(styleJson));
});
</script>

```

LINKED-STYLE.CS

Applying a style

The styles are applied using the **applyStyle** method of **editorModule**, the parameter should be passed in the **Name** of the Style.

The styles of the **Character** type is applied to the currently selected part of the document. If there is no selection, the values will be applied to the word at caret position. The styles of **Paragraph** type follows the same logic and are applied to all paragraphs in the selection or the current paragraph.

When there is no selection, styles of **Linked** type will change the values of the paragraph, and apply both the Paragraph and Character properties. When there is selection, Linked Style changes only the character properties of the selected text.

For example, the following line will apply the "New Linked" to the current paragraph.

```

`typescript
documentEditor.editor.applyStyle('New Linked');
//Clear direct formatting and apply the specified style
documentEditor.editor.applyStyle('New Linked', true);
`

```

Working with Lists

Document editor supports both the single-level and multilevel lists. Lists are used to organize data as step-by-step instructions in documents for easy understanding of key points. You can apply list to the paragraph either using supported APIs.

Create bullet list

Bullets are usually used for unordered lists. To apply bulleted list for selected paragraphs, use the following method of 'Editor' instance.

Note: applyBullet(bullet, fontFamily);

Parameter	Type	Description
bullet	string	Bullet character.
fontFamily	string	Bullet font family.

```
`typescript
```

```
documenteditor.editor.applyBullet("\uf0b7", 'Symbol');
```

```
,
```

Create numbered list

Numbered lists are usually used for ordered lists. To apply numbered list for selected paragraphs, use the following method of 'Editor' instance.

Note: applyNumbering(numberFormat,listLevelPattern)

Parameter	Type	Description
numberFormat	string	"%n" representations in 'numberFormat' parameter will be replaced by respective list level's value. "%1" will be displayed as "1"
listLevelPattern(optional)	string	Default value is 'Arabic'.

```
`typescript
```

```
documenteditor.editor.applyNumbering('%1', 'UpRoman');
```

```
,
```

Clear list

You can also clear the list formatting applied for selected paragraphs.

```
`typescript
```

```
documenteditor.editor.clearList();
```

```
,
```

Working with lists

CSHTML

```
@Html.EJS().Toolbar("defaultToolbar").Items(new List<ToolbarItem> {
    new ToolbarItem { PrefixIcon = "e-de-icon-Bullets", Id="Bullets",
    TooltipText = "Bullets" },
```

```

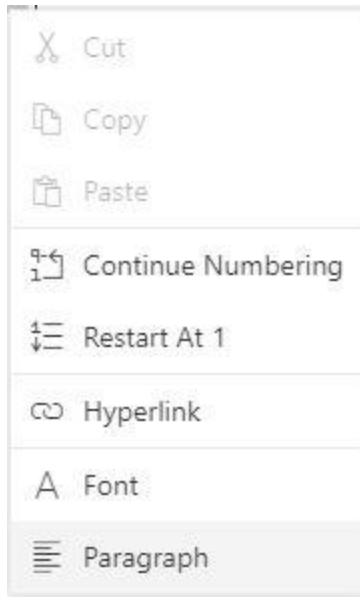
        new ToolbarItem { PrefixIcon = "e-de-icon-Numbering", Id="Numbering",
        TooltipText = "Numbering" },
        new ToolbarItem { Text = "Clear", Id="clearlist", TooltipText = "Clear
List" })).Clicked("toolbarAction").Render()
<div id="documenteditor">
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).EnableEditorHistory(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.resize();
        updateContainerSize();
        function toolbarAction(args) {
            switch (args.item.id) {
                case 'Bullets':
                    //To create bullet list
                    documenteditor.editor.applyBullet('\uf0b7', 'Symbol');
                    break;
                case 'Numbering':
                    //To create numbering list
                    documenteditor.editor.applyNumbering('%1', 'UpRoman');
                    break;
                case 'clearlist':
                    //To clear list
                    documenteditor.editor.clearList();
                    break;
            }
        };
    });
</script>

```

LIST.CS

Editing numbered list

Document editor restarts the numbering or continue numbering for a numbered list. These options are found in the built-in context menu, if the list value is selected.



See Also

- [List dialog](#)

Working with Table Formatting

Document editor customizes the formatting of table, or table cells such as table width, cell margins, cell spacing, background color, and table alignment. This section describes how to customize these formatting for selected cells, rows, or table in detail.

Cell margins

You can customize the cell margins by using the following sample code.

```
`typescript
//To change the left margin
documenteditor.selection.cellFormat.leftMargin=5.4;
//To change the right margin
documenteditor.selection.cellFormat.rightMargin=5.4;
//To change the top margin
documenteditor.selection.cellFormat.topMargin=5.4;
//To change the bottom margin
documenteditor.selection.cellFormat.bottomMargin=5.4;
`
```

You can also define the default cell margins for a table. If the specific cell margin value is not defined explicitly in the cell formatting, the corresponding value will be retrieved from default cells margin of the table.

```
`typescript
```

```
//To change the left margin
documenteditor.selection.tableFormat.leftMargin=5.4;

//To change the right margin
documenteditor.selection.tableFormat.rightMargin=5.4;

//To change the top margin
documenteditor.selection.tableFormat.topMargin=5.4;

//To change the bottom margin
documenteditor.selection.tableFormat.bottomMargin=5.4;
`typescript
```

Background color

You can explicitly set the background color of selected cells using the following sample code.

```
`typescript
documenteditor.selection.cellFormat.background='#E0E0E0';
`
```

Refer to the following sample code to customize the background color of the table.

```
`typescript
documenteditor.selection.tableFormat.background='#E0E0E0';
`
```

Cell spacing

Refer to the following sample code to customize the spacing between each cell in a table.

```
`typescript
documenteditor.selection.tableFormat.cellSpacing = 2;
`
```

Cell vertical alignment

The content is aligned within a table cell to **Top**, **Center**, or **Bottom**. You can customize this property of selected cells.

```
`typescript
documenteditor.selection.cellFormat.verticalAlignment= 'Bottom';
`
```

Table alignment

The tables are aligned in document editor to **Left**, **Right**, or **Center**.

```
`typescript
documenteditor.selection.tableFormat.tableAlignment='Center';
`
```

Cell width

Set the desired width of table cells that will be considered when the table is layouted.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.editor.insertTable(2, 2);
        //To change the width of a cell
        documenteditor.selection.cellFormat.preferredWidthType = 'Point';
        documenteditor.selection.cellFormat.preferredWidth = 100;
    });
</script>
```

CELL-WIDTH.CS

--

Table width

You can set the desired width of a table in **Point** or **Percent** type.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.editor.insertTable(2, 2);
        //To change the width of a cell
        documenteditor.selection.tableFormat.preferredWidthType = 'Point';
        documenteditor.selection.tableFormat.preferredWidth = 100;
    });
</script>
```

TABLE-WIDTH.CS

--

Apply borders

Document editor exposes API to customize the borders for table cells by specifying the settings.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
```

```

<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.editor.insertTable(2, 2);
        //To apply border
        var borderSettings = {
            type: 'AllBorders',
            lineWidth: 12
        };
        documenteditor.editor.applyBorders(borderSettings);
    });
</script>

```

APPLY-BORDERS.CS

--

Working with row formatting

Document editor allows various row formatting such as height and repeat header.

Row height

You can customize the height of a table row as **Auto**, **AtLeast**, or **Exactly**.

CSHTML

```

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.editor.insertTable(2, 2);
        //To change row height of first row
        documenteditor.selection.rowFormat.heightType = 'Exactly';
        documenteditor.selection.rowFormat.height = 20;
    });
</script>

```

ROW-HEIGHT.CS

--

Header row

The header row describes the content of a table. A table can optionally have a header row. Only the first row of a table can be the header row. If the cursor position is at first row of the table, then you can define whether it as header row or not, using the following sample code.

```
`typescript
```

```
documenteditor.selection.rowFormat.isHeader=true;
```


Allow row break across pages

This property is valid if a table row does not fit in the current page during table layout. It defines whether a table row can be allowed to break. If the value is false, the entire row will be moved to the start of next page. You can modify this property for selected rows using the following sample code.

```
`typescript
documenteditor.selection.rowFormat.allowRowBreakAcrossPages=false;
```

Title

Document Editor expose API to get or set the table title of the selected table. Refer to the following sample code to set title.

```
`typescript
documenteditor.selection.tableFormat.title = 'Shipping Details';
```

Description

Document Editor expose API to get or set the table description of the selected image. Refer to the following sample code to set description.

```
`typescript
documenteditor.selection.tableFormat.description = 'Freight cost and shipping details';
```

See Also

- [Table properties dialog](#)

Working with Section Formatting

Document editor supports various section formatting such as page size, page margins, and more.

Page size

You can get or set the size of a section at cursor position by using the following sample code.

```
`typescript
documenteditor.selection.sectionFormat.pageWidth = 500;
documenteditor.selection.sectionFormat.pageHeight = 600;
```

You can change the orientation of the page by swapping the values of page width and height respectively.

Page margins

Left and right page margin defines the gap between the document content from left and right side of the page respectively. Top and bottom page margins defines the gap between the document content from header and footer of the page respectively.

```
`typescript
documenteditor.selection.sectionFormat.leftMargin = 10;
documenteditor.selection.sectionFormat.rightMargin = 10;
documenteditor.selection.sectionFormat.bottomMargin = 10;
documenteditor.selection.sectionFormat.topMargin = 10;
`
```

Header distance

You can define the distance of header content from the top of the page by using the following sample code.

```
`typescript
documenteditor.selection.sectionFormat.headerDistance = 72;
`
```

Footer distance

You can define the distance of footer content from the bottom of the page by using the following sample code.

```
`typescript
documenteditor.selection.sectionFormat.footerDistance = 72;
`
```

See Also

- [Pagesetup dialog](#)

Comments in Document Editor Component

Document editor allows to add comments to documents. You can add, navigate and remove comments in code and from the UI.

Add a new comment

Comments can be inserted to the selected text.

```
`typescript
documentEditor.editor.insertComment("Test comment");
`
```

Comment navigation

Next and previous comments can be navigated using the below code snippet.

```
`typescript
//Navigate to next comment
documentEditor.selection.navigateNextComment();

//Navigate to previous comment
```

```
documentEditor.selection.navigatePreviousComment();
```

```
,
```

Delete comment

Current comment can be deleted using the below code snippet.

```
`typescript
```

```
documentEditor.editor.deleteComment();
```

```
,
```

Delete all comment

All the comments in the document can be deleted using the below code snippet.

```
`typescript
```

```
documentEditor.editor.deleteAllComments();
```

```
,
```

Protect the document in comments only mode

Document Editor provides support for protecting the document with **CommentsOnly** protection. In this protection, user is allowed to add or edit comments alone in the document.

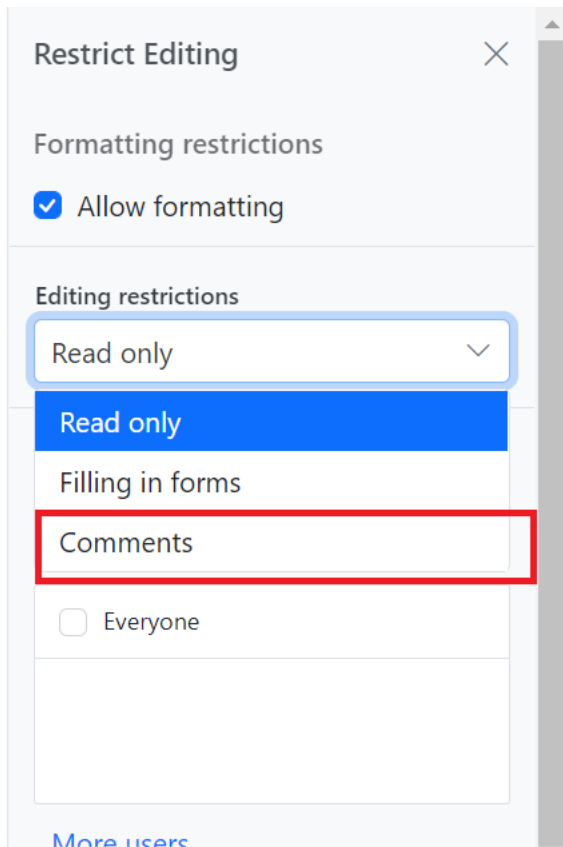
Document editor provides an option to protect and unprotect document using **enforceProtection** and **stopProtection** API.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById('container');
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        container.documentEditor.editor.enforceProtection('123',
        'CommentsOnly');
        //stop the document protection
        container.documentEditor.editor.stopProtection('123');
    }
</script>
```

COMMENT-ONLY.CS

Comment only protection can be enabled in UI by using [Restrict Editing pane](#)



Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are NoProtection | ReadOnly | FormFieldsOnly | CommentsOnly. In stop protection method, parameter denotes the password.

Fields

Document Editor has preservation support for all types of fields in an existing word document without any data loss.

Adding Fields

You can add a field to the document by using [insertField](#) method in **Editor** module.

```
`typescript
var fieldCode = 'MERGEFIELD First Name \* MERGEFORMAT';
var fieldResult = '«First Name»';
documenteditor.editor.insertField(fieldCode, fieldResult);
`
```

Note: Document editor does not validate or process the field code or field result. It simply inserts the field with specified field information.

Update fields

Document Editor provides support for updating bookmark cross reference field.

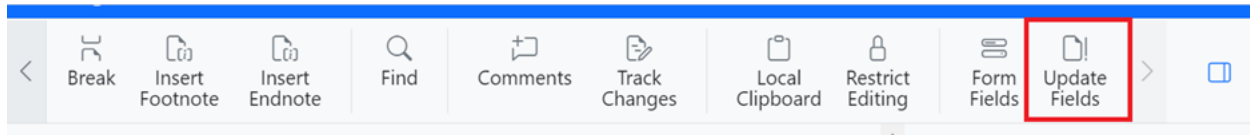
```
`typescript
```

//Update all the bookmark cross reference field in the document.

```
documenteditor.updateFields();
```

,

Bookmark cross reference fields can be updated through UI by using update fields option in **Toolbar**.



The following type of fields are automatically updated in Document Editor.

- Numpages
- Section
- Page

Get field info

You can get field code and field result of the current selected field by using [getFieldInfo](#) method in the **Selection** module.

```
`typescript
```

```
//Gets the field information of the selected field.
```

```
var fieldInfo = documenteditor.selection.getFieldInfo();
```

,

Note: For nested fields, this method returns combined field code and result.

Set field info

You can modify the field code and field result of the current selected field by using [setFieldInfo](#) method in the **Editor** module.

```
`typescript
```

```
//Gets the field information for the selected field.
```

```
var fieldInfo = documenteditor.selection.getFieldInfo();
```

```
//Modify field code
```

```
fieldInfo.code = 'MERGEFIELD First Name \* MERGEFORMAT ';
```

```
//Modify field result
```

```
fieldInfo.result = '«First Name»';
```

```
//Modify field code and result of the current selected field.
```

```
documenteditor.editor.setFieldInfo(fieldInfo);
```

,

Note: For nested field, entire field gets replaced completely with the specified field information.

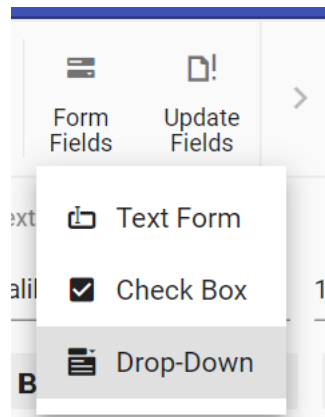
See Also

[Mail merge using DocIO](#)

[Mail merge demo](#)

Form Fields in Document Editor Control

DocumentEditorContainer control provides support for inserting Text, CheckBox, DropDown form fields through in-built toolbar.



Insert form field

Form fields can be inserted using `insertFormField` method in editor module.

`typescript

```
//Insert Text form field
```

```
documentEditor.editor.insertFormField('Text');
```

```
//Insert Checkbox form field
```

```
documentEditor.editor.insertFormField('CheckBox');
```

```
//Insert Drop down form field
```

```
documentEditor.editor.insertFormField('Dropdown');
```

```
,
```

Get form field names

All the form fields names from current document can be retrieved using `getFormFieldNames()`.

`typescript

```
var formFieldsNames = documentEditor.getFormFieldNames();
```

```
,
```

Get form field properties

Form field properties can be retrieved using `getFormFieldInfo()`.

`typescript

```
//Text form field
```

```
var textfieldInfo = documentEditor.getFormFieldInfo('Text1');
```

```
//Checkbox form field
```

```
var checkboxfieldInfo = documentEditor.getFormFieldInfo('Check1');
```

```
//Dropdown form field
```

```
var dropdownfieldInfo = documentEditor.getFormFieldInfo('Drop1');
```

```
,
```

Set form field properties

Form field properties can be modified using `setFormFieldInfo`.

```
`typescript
```

```
// Set text form field properties
```

```
var textfieldInfo = documentEditor.getFormFieldInfo('Text1');
```

```
textfieldInfo.defaultValue = "Hello";
```

```
textfieldInfo.format = "Uppercase";
```

```
textfieldInfo.type = "Text";
```

```
documentEditor.setFormFieldInfo('Text1',textfieldInfo);
```

```
// Set checkbox form field properties
```

```
var checkboxfieldInfo = documentEditor.getFormFieldInfo('Check1');
```

```
checkboxboxfieldInfo.defaultValue = true;
```

```
documentEditor.setFormFieldInfo('Check1',checkboxboxfieldInfo);
```

```
// Set checkbox form field properties
```

```
var dropdownfieldInfo = documentEditor.getFormFieldInfo('Drop1');
```

```
dropdownfieldInfo.dropDownItems = ['One','Two', 'Three']
```

```
documentEditor.setFormFieldInfo('Drop1',dropdownfieldInfo);
```

```
,
```

Export form field data

Data of the all Form fields in the document can be exported using `exportFormData`.

```
`typescript
```

```
var formFieldDate = documentEditor.exportFormData();
```

```
,
```

Import form field data

Form fields can be prefilled with data using `importFormData`.

```
`typescript
```

```
var textformField = {fieldName: 'Text1', value: 'Hello World'};
```

```
var checkformField = {fieldName: 'Check1', value: true};
```

```
var dropdownformField = {fieldName: 'Drop1', value: 1};
```

//Import form field data

```
documentEditor.importFormData([textFormField,checkFormField,dropdownFormField]);
```

,

Reset form fields

Reset all the form fields in current document to default value using `resetFormFields`.

`typescript

```
documentEditor.resetFormFields();
```

,

Protect the document in form filling mode

Document Editor provides support for protecting the document with `FormFieldsOnly` protection. In this protection, user can only fill form fields in the document.

Document editor provides an option to protect and unprotect document using `enforceProtection` and `stopProtection` API.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById('container');
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        container.documentEditor.editor.enforceProtection('123',
'FormFieldsOnly');
        //stop the document protection
        container.documentEditor.editor.stopProtection('123');
    }
</script>
```

PROTECT-UNPROTECT.CS

Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are `NoProtection` | `ReadOnly` | `FormFieldsOnly` | `CommentsOnly`. In stop protection method, parameter denotes the password.

Clipboard

Document editor takes advantage of system clipboard and allows to copy or move a portion of the document into it in HTML format, so that it can be pasted in any application that supports clipboard.

Copy

Copy a portion of document to system clipboard using built-in context menu of document editor. You can also do it programmatically using the following sample code.


```
`typescript
```

```
documentEditor.selection.copy();
```

```
,
```

Cut

Cut a portion of document to system clipboard using built-in context menu of document editor. You can also do it programmatically using the following sample code.

```
`typescript
```

```
documentEditor.editor.cut();
```

```
,
```

Paste

Due to limitations, you can paste contents from system clipboard as plain text in document editor only using the 'CTRL + V' keyboard shortcut.

Local paste

Document editor exposes API to enable local paste within the control. On enabling this, the following is performed:

- Selected contents will be stored to an internal clipboard in addition to system clipboard.
- Clipboard paste will be overridden, and internally stored data that has formatted text will be pasted.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.enableLocalPaste = true;
    });
</script>
```

CLIPBOARD.CS

By default, **enableLocalPaste** is false. When local paste is enabled for a document editor instance, you can paste contents programmatically if the internal clipboard has stored data during last copy operation.

```
`typescript
```

```
documentEditor.editor.pasteLocal();
```

```
,
```

EnableLocalPaste behaviour

|EnableLocalPaste |Paste behavior details|

|-----|-----|

|True |Allows to paste content that is copied from the same Document editor component alone and prevents pasting content from system clipboard. Hence the content copied from outside Document editor component can't be pasted.
Browser limitation of pasting from system clipboard using API and context menu options, will be resolved. So, you can copy and paste content within the Document editor component using API and context menu options too. |

|False |Allows to paste content from system clipboard. Hence the content copied from both the Document editor component and outside can be pasted.
Browser limitation of pasting from system clipboard using API and context menu options, will remain as a limitation. |

Note: Keyboard shortcut for pasting will work properly in both cases. Copying content from Document editor component and pasting outside will work properly in both cases.

Paste with formatting

Document Editor provides support to paste the system clipboard data with formatting. To enable clipboard paste with formatting options, set the `EnableLocalPaste` property in Document Editor to false and use this .NET Standard library [Synconfusion.EJ2.WordEditor.AspNet.Core](#) by the web API service implementation. This library helps to paste the system clipboard data with formatting.

You can paste your system clipboard data in the following ways:

- **Keep Source Formatting** This option retains the character styles and direct formatting applied to the copied text. Direct formatting includes characteristics such as font size, italics, or other formatting that is not included in the paragraph style.
- **Match Destination Formatting** This option discards most of the formatting applied directly to the copied text, but it retains the formatting applied for emphasis, such as bold and italic when it is applied to only a portion of the selection. The text takes on the style characteristics of the paragraph where it is pasted. The text also takes on any direct formatting or character style properties of text that immediately precedes the cursor when the text is pasted.
- **Text Only** This option discards all formatting and non-text elements such as pictures or tables. The text takes on the style characteristics of the paragraph where it is pasted and takes on any direct formatting or character style properties of text that immediately precedes the cursor when the text is pasted. Graphical elements are discarded and tables are converted to a series of paragraphs.

This paste option appears as follows.

Adventure Works Cycles

Hello world!

Adventure Works Cycles, the fictitious company on which the Adventure Works sample databases are based, manufactures and sells mountain, hybrid, and commercial bicycles in several regional sales teams. In 2000, Adventure Works was located in Mexico. Importadores Neptuno, a company in Bothell, Washington, became the sole manufacturer and distributor of the Adventure Works Cycles product line. These subcomponents are shipped to the Bothell location for final product assembly. In 2001, Importadores Neptuno, became the sole manufacturer and distributor of the touring bicycle product group.

See Also

- [Keyboard shortcuts](#)

History

Document editor tracks the history of all editing actions done in the document, which allows undo and redo functionality.

Enable or disable history

Inject the `EditorHistory` module in your application to provide history preservation functionality for `DocumentEditor`.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSfdtExport(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
    document.getElementById("container").ej2_instances[0];
        documenteditor.enableEditorHistory = true;
    });
</script>
```

HISTORY.CS

You can enable or disable history preservation for a document editor instance any time using the `enableEditorHistory` property.

```
`typescript
```

```
editor.enableEditorHistory = false;
```

Undo and redo

You can perform undo and redo by **CTRL+Z** and **CTRL+Y** keyboard shortcuts. Document editor exposes API to do it programmatically. To undo the last editing operation in document editor, refer to the following sample code.

```
`typescript
editor.editorHistory.undo();
```

To redo the last undone action, refer to the following code example.

```
`typescript
editor.editorHistory.redo();
```

Stack size

History of editing actions will be maintained in stack, so that the last item will be reverted first. By default, document editor limits the size of undo and redo stacks to 500 each respectively. However, you can customize this limit.

```
`typescript
editor.editorHistory.undoLimit = 400;
editor.editorHistory.redoLimit = 400;
```

See Also

- [Feature modules](#)
- [Keyboard shortcuts](#)

Find and Replace in Document Editor Component

The document editor component searches a portion of text in the document through a built-in interface called **OptionsPane** or rich APIs. When used in combination with selection performs various operations on the search results like replacing it with some other text, highlighting it, making it bolder, and more.

Options pane

This provides the options to search for a portion of text in the document. After search operation is completed, the search results will be displayed in a list and options to navigate between them. The current occurrence of matched text or all occurrences with another text can be replaced by switching to **Replace** tab. This pane is opened using the keyboard shortcut **CTRL+F**.

CSHTML

```
@Html.EJS().Button("showhidepane").Content("Show hide pane").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").EnableSelection(true).EnableSearch(true).IsReadOnly(false).EnableEditor(true).EnableOptionsPane(true).Render()
```

```
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var sfdt = {
            "sections": [
                {
                    "blocks": [
                        {
                            "inlines": [
                                {
                                    "characterFormat": {
                                        "bold": true,
                                        "italic": true
                                    },
                                    "text": "Adventure Works Cycles, the
fictitious company on which the AdventureWorks sample databases are based,
is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Bothell,
Washington with 290 employees, several regional sales teams are located
throughout their market base."
                                }
                            ]
                        }
                    ]
                }
            ]
        };
        documenteditor.open(JSON.stringify(sfdt));
        document.getElementById('showhidepane').addEventListener('click',
function () {
            documenteditor.showOptionsPane();
        });
    });
</script>
```

OPTIONS-PANE.CS

You can close the options pane by pressing **Esc** key.

Search

The **Search** module of Document Editor exposes the following APIs:

API Name	Type	Description
----------	------	-------------

---	---	---
-----	-----	-----

findAll()	Method	Searches for specified text in the whole document and highlights it with yellow.
------------------	--------	--

searchResults	Property	This is an instance of SearchResults .
----------------------	----------	---

find() | Method | Find immediate occurrence of specified text from cursor position in the document and highlights it with yellow. |

Find the immediate occurrence in the document

Using **find()** method, you can find the immediate occurrence of specified text from current cursor position in the document.

`typescript

```
documenteditor.search.find('Some text', 'None');
```

,

Note: Second parameter is optional parameter and it denotes find Options. Possible values of find options are 'None' | 'WholeWord' | 'CaseSensitive' | 'CaseSensitiveWholeWord'.

Find all the occurrences in the document

Using **findAll()** method, you can find all the occurrences of specified text in the whole document and highlight it with yellow.

`typescript

```
documenteditor.search.findAll('Some text', 'None');
```

,

Note: Second parameter is optional parameter and it denotes to find Options. Possible values of find options are 'None' | 'WholeWord' | 'CaseSensitive' | 'CaseSensitiveWholeWord'.

Search results

The **SearchResults** class provides information about the search results after search operation is completed that can be identified using the **searchResultsChange** event. This will expose the following APIs:

| API Name | Type | Description |

| --- | --- | --- |

| **length** | Property | Returns the total number of results found on the search. |

| **index** | Property | Returns the index of selected search result. You can change the value for this property to move the selection. |

| **replaceAll()** | Method | Replaces all the occurrences with specified text. |

| **clear()** | Method | Clears the search result. |

Replace all the occurrences

Using **replaceAll**, you can replace all the occurrences with specified text.

`typescript

```
documentEditor.search.findAll ('Some text');
```

```
// Replace all the searched text with word 'Mike'
```

```
documentEditor.search.searchResults.replaceAll("Mike");
```

,

Replace

Using `insertText`, you can replace the current searched text with specified text and it replaces single occurrence.

Note: Note: This `insertText` API accepts following control characters.

`
`* New line characters ("`\r`", "`\r\n`", "`\n`") - Inserts a new paragraph and appends the remaining text to the new paragraph.

`
`* Line break character ("`\v`") - Moves the remaining text to start in new line.

`
`* Tab character ("`\t`") - Allocates a tab space and continue the next character.

`typescript

```
container.documentEditor.search.findAll('works');
let searchLength: number = container.documentEditor.search.searchResults.length;
for (let i = 0; i < searchLength; i++) {
  // It will move selection to specific searched index,move to each occurrence one by one
  container.documentEditor.search.searchResults.index = i;
  // Replace it with some text
  container.documentEditor.editor.insertText('Hello');
}
container.documentEditor.search.searchResults.clear();
`
```

SearchResultsChange event

`DocumentEditor` exposes the `searchResultsChange` event that will be triggered whenever search results are changed. Consider the following scenarios:

- A search operation is completed with some results.
- The results are replaced with some other text, since it will be cleared automatically.
- The results are cleared explicitly.

`typescript

```
documenteditor.searchResultsChange = function() {
};
`
```

Customize find and replace

Using the exposed APIs, you can customize the find and replace functionality in your application.

CSHTML

```
@Html.EJS().Button("replace_all").Content("Replace All").Render()
<div id="documenteditor" style="width:100%;height:100%">
```

```

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSearch(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var sfdt = {
            "sections": [
                {
                    "blocks": [
                        {
                            "inlines": [
                                {
                                    "characterFormat": {
                                        "bold": true,
                                        "italic": true
                                    },
                                    "text": "Adventure Works Cycles, the
fictitious company on which the AdventureWorks sample databases are based,
is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Bothell,
Washington with 290 employees, several regional sales teams are located
throughout their market base."
                                }
                            ]
                        }
                    ]
                }
            ]
        };
        documenteditor.open(JSON.stringify(sfdt));
        document.getElementById('replace_all').addEventListener('click',
function () {
            var textToFind = document.getElementById('find_text').value;
            var textToReplace =
document.getElementById('replace_text').value;
            if (textToFind !== '') {
                // Find all the occurrences of given text
                documenteditor.searchModule.findAll(textToFind);
                if (documenteditor.searchModule.searchResults.length > 0) {
                    // Replace all the occurrences of given text

documenteditor.searchModule.searchResults.replaceAll(textToReplace);
                }
            }
        });
    });
</script>

```

FIND-REPLACE.CS

See Also

- [Options pane](#)
- [Feature modules](#)

Keyboard Shortcuts in Document Editor Component

Text formatting

The following table lists the default keyboard shortcuts in document editor for formatting text:

Key combination	Description
----- -----	
Ctrl + B	Toggles the bold property of selected text.
Ctrl + I	Toggles the italic property of selected text.
Ctrl + U	Toggles the underline property of selected text.
Ctrl + +	Toggles the subscript formatting of selected text.
Ctrl + Shift + +	Toggles the superscript formatting of selected contents.
Ctrl + }	Increases the actual font size of selected text by one point.
Ctrl + {	Decreases the actual font size of selected text by one point.

Paragraph formatting

The following table lists the default keyboard shortcuts for formatting the paragraph:

Key combination	Description
----- -----	
Ctrl + E	Selected paragraphs are center aligned.
Ctrl + J	Selected paragraphs are justified.
Ctrl + L	Selected paragraphs are left aligned.
Ctrl + R	Selected paragraphs are right aligned.
Ctrl + 1	Single line spacing is applied for selected paragraphs.
Ctrl + 5	1.5 line spacing is applied for selected paragraphs.
Ctrl + 2	Double spacing is applied for selected paragraphs.
Ctrl + 0	No spacing is applied before the selected paragraphs.
Ctrl + M	Increases the left indent of selected paragraphs by a factor of 36 points.
Ctrl + Shift + M	Decreases the left indent of selected paragraphs by a factor of 36 points.
Ctrl + *	Show/Hide the hidden characters like spaces, tab, paragraph marks, and breaks.

Clipboard

Key Combination	Description
-----------------	-------------

-----	-----
-------	-------

Ctrl + C	Copies selected contents to the clipboard.
----------	--

Ctrl + V	Pastes plain text content from the clipboard.
----------	---

Ctrl + X	Moves selected content to the clipboard.
----------	--

Keyboard shortcut to navigate around the document

Key Combination	Description
-----------------	-------------

-----	-----
-------	-------

Left arrow	Moves the cursor position one character to the left.
------------	--

Right arrow	Moves the cursor position one character to the right.
-------------	---

Down arrow	Moves the cursor position down one line.
------------	--

Up arrow	Moves the cursor position up one line.
----------	--

Ctrl + Left arrow	Moves the cursor position one word to the left.
-------------------	---

Ctrl + Right arrow	Moves the cursor position one word to the right.
--------------------	--

Ctrl + Up arrow	Moves the cursor position one paragraph up.
-----------------	---

Ctrl + Down arrow	Moves the cursor position one paragraph down.
-------------------	---

Tab (in table)	Moves the cursor position one cell to the right.
----------------	--

Shift + Tab (in table)	Moves the cursor position one cell to the left.
------------------------	---

Home	Moves the cursor position to the start of a line.
------	---

End	Moves the cursor position to the end of a line.
-----	---

Page up	Moves the cursor position one screen up.
---------	--

Page down	Moves the cursor position one screen down.
-----------	--

Ctrl + Home	Moves the cursor position to the start of a document.
-------------	---

Ctrl + End	Moves the cursor position to the end of a document.
------------	---

Keyboard shortcut to extend selection

Key Combination	Description
-----------------	-------------

-----	-----
-------	-------

Shift + Left arrow	Extends selection one character to the left.
--------------------	--

Shift + Right arrow	Extends selection one character to the right.
---------------------	---

Shift + Down arrow	Extends selection one line downward.
--------------------	--------------------------------------

Shift + Up arrow	Extends selection one line upward.
------------------	------------------------------------

Shift + Home	Extends selection to the start of a line.
--------------	---

Shift + End	Extends Selection to the end of a line.
-------------	---

Ctrl + A	Extends selection to the entire document.
Ctrl + Shift + Left arrow	Extends selection one word to the left.
Ctrl + Shift + Right arrow	Extends selection one word to the right.
Ctrl + Shift + Down arrow	Extends selection to the end of a paragraph.
Ctrl + Shift + Up arrow	Extends selection to the start of a paragraph.
Ctrl + Shift + Home	Extends selection to the start of a document.
Ctrl + Shift + End	Extends selection to the end of a document.

Find and Replace

Key Combination	Description
----- -----	
Ctrl + F	Opens options pane.
Ctrl + H	Opens replace tab in options pane.

Create, Save and Print document

Key Combination	Description
----- -----	
Ctrl + N	Opens empty document.
Ctrl + S	Saves the document in SFDT format.
Ctrl + P	Prints the document.

Edit Operation

Key Combination	Description
----- -----	
Backspace	Deletes one character to the left.
Delete	Deletes one character to the right.
Ctrl + Z	Undo last performed action.
Ctrl + Y	Redo last undo action.

Insert special characters

Key Combination	Description
----- -----	
Ctrl + Enter	Inserts page break.
Shift + Enter	Inserts line break.

Dialog

Key Combination	Description
----- -----	
Ctrl + F	Opens options pane.

|Ctrl + D| Opens font dialog.|

|Ctrl + K| Opens hyperlink dialog.|

See Also

- [How to override the keyboard shortcuts.](#)

Scrolling

The Document editor renders the document page by page. You can scroll through the pages by mouse wheel or touch interactions. You can also scroll through the page by using 'scrollToPage()' method of document editor instance.

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">
    @Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
        onLoadDefault();
        documenteditor.scrollToPage(2);
    });
    function onLoadDefault() {
        var defaultDocument = {
            "sections": [
                {
                    "blocks": [
                        {
                            "paragraphFormat": {
                                "styleName": "Normal"
                            },
                            "inlines": [
                                {
                                    "text": "First page"
                                }
                            ]
                        }
                    ],
                    "headersFooters": {},
                },
                {
                    "blocks": [
                        {
                            "paragraphFormat": {
                                "styleName": "Normal"
                            },
                            "inlines": [
                                {
                                    "text": "Second page"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    }
}
```

```

        },
        "headersFooters": {},
    },
    "characterFormat": {},
    "paragraphFormat": {},
    "background": {
        "color": "#FFFFFF"
    },
    "styles": [
        {
            "type": "Paragraph",
            "name": "Normal",
            "next": "Normal"
        },
        {
            "type": "Character",
            "name": "Default Paragraph Font"
        }
    ]
}
documenteditor.open(JSON.stringify(defaultDocument));
documenteditor.focusIn();
}
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

Note: Calling this method brings the specified page into view but doesn't move selection. Hence this method will work by default. That is, it works even if selection is not enabled.

In case, if you wish to move the selection to any page in document editor and bring it into view, you can use 'goToPage()' method of selection instance.

CSHTML

```

<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableSelecti
on(true).EnableSfdtExport(true).EnableEditor(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
        onLoadDefaultDocument();
        documenteditor.viewer.selection.goToPage(3);
    });
</script>

```

```
});  
function onLoadDefaultDocument() {  
    var defaultDocument = {  
        "sections": [  
            {  
                "blocks": [  
                    {  
                        "paragraphFormat": {  
                            "styleName": "Normal"  
                        },  
                        "inlines": [  
                            {  
                                "text": "First page"  
                            }  
                        ]  
                    }  
                ],  
                "headersFooters": {},  
            },  
            {  
                "blocks": [  
                    {  
                        "paragraphFormat": {  
                            "styleName": "Normal"  
                        },  
                        "inlines": [  
                            {  
                                "text": "Second page"  
                            }  
                        ]  
                    }  
                ],  
                "headersFooters": {},  
            },  
            {  
                "blocks": [  
                    {  
                        "paragraphFormat": {  
                            "styleName": "Normal"  
                        },  
                        "inlines": [  
                            {  
                                "text": "Third page"  
                            }  
                        ]  
                    }  
                ],  
                "headersFooters": {},  
            }  
        ],  
        "characterFormat": {},  
        "paragraphFormat": {},  
        "background": {  
            "color": "#FFFFFF"  
        },  
        "styles": [  
            {
```

```

        "type": "Paragraph",
        "name": "Normal",
        "next": "Normal"
    },
    {
        "type": "Character",
        "name": "Default Paragraph Font"
    }
]
};
documenteditor.open(JSON.stringify(defaultDocument));
documenteditor.focusIn();
}
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

Zooming

You can scale the contents in document editor ranging from 10% to 500% of the actual size. You can achieve this using mouse or touch interactions. You can also use 'zoomFactor' property of document editor instance. The value can be specified in a range from 0.1 to 5.

CSHTML

```

<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableSelecti
on(true).EnableSfdtExport(true).EnableEditor(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
        documenteditor.zoomFactor = 3;
    });
</script>

```

SCROLLING-ZOOMING.CS

Page Fit Type

Apart from specifying the zoom factor as value, the Document editor provides option to specify page fit options such as fit to full page or fit to page width. You can set this option using 'fitPage' method of document editor instance.

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableSelecti
on(true).EnableSfdtExport(true).EnableEditor(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
        documenteditor.fitPage('FitPageWidth');
    });
</script>
```

SCROLLING-PAGE-FIT.CS

Zoom option using UI

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableSelecti
on(true).EnableSfdtExport(true).EnableEditor(true).Render()
</div>
<div id='page-fit-type-div'>
    <label style="margin-top: 6px;margin-right: 2px">Page</label>
    <div id='editablePageNumber' style='border: 1px solid #F1F1F1;display:
inline-flex;height: 17px;padding: 0px 4px;'>
        <label id="documenteditor_page_no" style="text-
transform:capitalize;white-space:pre;overflow:hidden;user-
select:none;cursor:text;height:17px;max-width:150px"></label>
    </div>
    <label style="margin-left:2px;letter-spacing: 1.05px;">of</label>
    <label id='documenteditor_pagecount' style="margin-left:6px;letter-
spacing: 1.05px;"></label>
    @Html.EJS().DropDownButton("zoom").Content("100%).Items((IEnumerable
<object>) ViewBag.zoomList).Render()
</div>
<script>
    var documenteditor;
    var pageCount;
    var editablePageNumber;
    var editorPageCount;
    var pageNumberLabel;
```



```

var startPage = 1;
var zoomBtn;
document.addEventListener('DOMContentLoaded', function () {
    documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
    pageCount = document.getElementById('documenteditor_pagecount');
    editablePageNumber = document.getElementById('editablePageNumber');
    pageNumberLabel = document.getElementById('documenteditor_page_no');
    documenteditor.resize();
    editablePageNumber.addEventListener('click',
updateDocumentEditorPageNumber);
    editablePageNumber.addEventListener('keydown', onKeyDown);
    editablePageNumber.addEventListener('blur', onBlur);
    zoomBtn = document.getElementById('zoom');
    zoomBtn.className = 'e-de-statusbar-zoom';
    var zoom = zoomBtn.ej2_instances[0];
    zoom.select = function (e) {
        onZoom(e);
    };
    updatePageCount();
    updatePageNumber();
    editablePageNumber.addEventListener('click',
updateDocumentEditorPageNumber);
    editablePageNumber.addEventListener('keydown', onKeyDown);
    editablePageNumber.addEventListener('blur', onBlur);
    documenteditor.viewChange = function (e) {
        updatePageNumberOnViewChange(e);
    };
    documenteditor.contentChange = function () {
        //Set page count
        updatePageCount();
    };
});
function updatePageNumberOnViewChange(args) {
    if (documenteditor.selection
        && documenteditor.selection.startPage >= args.startPage &&
documenteditor.selection.startPage <= args.endPage) {
        startPage = documenteditor.selection.startPage;
    } else {
        startPage = args.startPage;
    }
    updatePageNumber();
}
function onBlur() {
    if (editablePageNumber.textContent === '' ||
parseInt(editablePageNumber.textContent, 0) > editorPageCount) {
        updatePageNumber();
    }
    editablePageNumber.contentEditable = 'false';
}
function onKeyDown(e) {
    if (e.which === 13) {
        e.preventDefault();
        var pageNumber = parseInt(editablePageNumber.textContent, 0);
        if (pageNumber > editorPageCount) {
            updatePageNumber();
        } else {

```

```

        if (documenteditor.selection) {
documenteditor.selection.goToPage(parseInt(editablePageNumber.textContent,
0));
        } else {
documenteditor.scrollToPage(parseInt(editablePageNumber.textContent, 0));
        }
        editablePageNumber.contentEditable = 'false';
        if (editablePageNumber.textContent === '') {
            updatePageNumber();
        }
    }
    if (e.which > 64) {
        e.preventDefault();
    }
}
function onZoom(args) {
    setZoomValue(args.item.text);
    updateZoomContent();
}
function setZoomValue(text) {
    if (text.match('Fit one page')) {
        documenteditor.fitPage('FitOnePage');
    } else if (text.match('Fit page width')) {
        documenteditor.fitPage('FitPageWidth');
    } else {
        documenteditor.zoomFactor = parseInt(text, 0) / 100;
    }
}
function updateZoomContent() {
    zoomBtn.content = Math.round(documenteditor.zoomFactor * 100) + '%';
}
function updatePageNumber() {
    pageNumberLabel.textContent = startPage.toString();
}
function updatePageCount() {
    editorPageCount = documenteditor.pageCount;
    pageCount.textContent = editorPageCount.toString();
}
function updateDocumentEditorPageNumber() {
    var editablePageNumber =
document.getElementById('editablePageNumber');
    editablePageNumber.contentEditable = 'true';
    editablePageNumber.focus();
    window.getSelection().selectAllChildren(editablePageNumber);
}
</script>
<style>
    #DocumentEditor {
        width: 100%;
        height: 100%;
    }
    .e-de-statusbar-zoom {
        float: right;
        text-align: center;

```

```
padding: 2px;
line-height: 19px;
margin-top: 1px;
}
</style>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    List<object> zoomItems = new List<object>();
    zoomItems.Add(new { text = "200%" });
    zoomItems.Add(new { text = "175%" });
    zoomItems.Add(new { text = "150%" });
    zoomItems.Add(new { text = "125%" });
    zoomItems.Add(new { text = "100%" });
    zoomItems.Add(new { text = "75%" });
    zoomItems.Add(new { text = "50%" });
    zoomItems.Add(new { text = "25%" });
    zoomItems.Add(new { separator = true });
    zoomItems.Add(new { text = "Fit one page" });
    zoomItems.Add(new { text = "Fit page width" });
    ViewBag.zoomList = zoomItems;
    return View();
}
```

Print in Document Editor Control

To print the document, use the `print` method from document editor instance.

CSHTML

```
@Html.EJS().Button("print").Content("Print").Render()
<div id="documenteditor" style="width:100%;height:100%">
    @Html.EJS().DocumentEditor("container").EnablePrint(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement =
document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var sfdt = {
            "sections": [
                {
                    "blocks": [
                        {
                            "inlines": [
                                {
                                    "characterFormat": {
                                        "bold": true,
                                        "italic": true
                                    },
                                    "text": "Hello World"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    });
</script>
```

```

        ]
        },
        "headersFooters": {
        }
    }
    ];
    documenteditor.open(JSON.stringify(sfdt));
    document.getElementById('print').addEventListener('click',
function () {
    documenteditor.print();
});
});
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

CSHTML

```

@Html.EJS().Button("print").Content("Print").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnablePrint(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement =
document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        document.getElementById('print').addEventListener('click',
function () {
            documenteditor.print();
        });
    });
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

Note: To enable print for a document editor instance, set `enablePrint` as `true`.

Improve print quality

Document editor provides an option to improve the print quality using `printDevicePixelRatio` in Document editor settings. Document editor using canvas approach to render content. Then, canvas are converted to image and it is processed for print. Using `printDevicePixelRatio` API, you can increase the image quality based on your requirement.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").EnableToolbar(true).DocumentEditorSettings("settings").Render()
<script>
    var settings = { printDevicePixelRatio : 2 };
</script>
```

PRINT.CS

Note: By default, `printDevicePixelRatio` value is 1.

Print using window object

You can print the document in document editor by passing the window instance. This is useful to implement print in third party frameworks such as electron, where the window instance will not be available.

CSHTML

```
@Html.EJS().Button("print").Content("Print").Render()
<div id="documenteditor">
    @Html.EJS().DocumentEditor("container").EnablePrint(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.print(window);
    });
</script>
```

PRINT-WINDOW.CS

Page setup

Some of the print options cannot be configured using JavaScript.

- [Chrome](#)
- [Firefox](#)

However, you can customize margins, paper, and layout options by modifying the section format properties using page setup dialog.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnablePrint(true).EnableSelection(true).EnableSfdtExport(true).EnablePageSe
tupDialog(true).Render()
<script>
    var documenteditor;
    var documenteditorElement = document.getElementById("container");
    documenteditorElement.style.height = "100%";
    documenteditorElement.style.width = "100%";
    documenteditor = documenteditorElement.ej2_instances[0];
    documenteditor.resize();
    documenteditor.showPageSetupDialog();
</script>
```

PRINT-DIALOG.CS

By customizing margins, papers, and layouts, the layout of the document will be changed in document editor. To modify these options during print operation, serialize the document as SFDT using the `serialize` method in document editor instance and open the SFDT data in another instance of document editor in separate window.

CSHTML

```
@Html.EJS().Button("print").Content("Print").Render()
<div id="documenteditor">

@Html.EJS().DocumentEditor("DocumentEditor1").IsReadOnly(false).EnableEditor
(true).EnableSelection(true).EnableSfdtExport(true).EnablePrint(true).Render
()

@Html.EJS().DocumentEditor("DocumentEditor2").IsReadOnly(false).EnableEditor
(true).EnableSelection(true).EnableSfdtExport(true).EnablePrint(true).Render
()
</div>
<script>
    var documenteditor1;
    var documenteditor2;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor1 =
document.getElementById('DocumentEditor1').ej2_instances[0];
        documenteditor2 =
document.getElementById('DocumentEditor2').ej2_instances[0];
        documenteditor1.resize();
        documenteditor2.resize();
    });
    document.getElementById('print').addEventListener('click', function () {
        var sfdtData = documenteditor1.serialize();
        documenteditor2.open(sfdtData);
        //Set A5 paper size
    });
</script>
```

```
documenteditor2.selection.sectionFormat.pageWidth = 419.55;
documenteditor2.selection.sectionFormat.pageHeight = 595.30;
documenteditor2.print();
});
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

See Also

- [Feature modules](#)
- [Page Setup dialog](#)

Dialog in Document Editor Component

Documenteditor provides dialog support to major operations such as insert or edit hyperlink, formatting text, paragraph, style, list and table properties.

Font Dialog

Font dialog allows to modify all text properties for selected contents at once such as bold, italic, underline, font size, font color, strikethrough, subscript and superscript.

Note: To enable font dialog for a document editor instance, set 'enableFontDialog' to true.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableFontDialog(true).EnableSelection(true).EnableSfdtExport(true).Render(
)
</div>
<script>
    var documenteditor;
    var containerPanel;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open Font Dialog
            documenteditor.showDialog('Font');
        });
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Paragraph dialog

This dialog allows modifying the paragraph formatting for selection at once such as text alignment, indentation, and spacing.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableParagraphDialog(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    var containerPanel;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open Paragraph Dialog
            documenteditor.showDialog('Paragraph');
        });
    });
</script>
```

PARAGRAPH-DIALOG.CS**Table dialog**

This dialog allows creating and inserting a table at cursor position by specifying the required number of rows and columns.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableTableDialog(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
```



```

var documenteditor;
document.addEventListener('DOMContentLoaded', function () {
    documenteditor =
document.getElementById("container").ej2_instances[0];
    var button = document.getElementById('dialog');
    button.addEventListener('click', function () {
        // To open Table Dialog
        documenteditor.showDialog('Table');
    });
});
</script>

```

TABLE-DIALOG.CS

Bookmark dialog

This dialog allows to perform the following operations:

- View all bookmarks.
- Navigate to a bookmark.
- Create a bookmark at current selection.
- Delete an existing bookmark.

CSHTML

```

@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableBookmarkDialog(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    var containerPanel;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open Bookmark Dialog
            documenteditor.showDialog('Bookmark');
        });
    });
</script>

```

BOOKMARK-DIALOG.CS

Hyperlink dialog

This dialog allows editing or inserting a hyperlink at cursor position.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableEditor(
true).EnableSelection(true).EnableSfdtExport(true).EnableHyperlinkDialog(tru
e).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
    });
    //Click the Dialog button, the hyperlink dialog will open
    document.getElementById('dialog').addEventListener('click', function ()
{
        documenteditor.showDialog('Hyperlink');
    });
</script>
```

HYPERLINK-DIALOG.CS

Table of contents dialog

This dialog allows creating and inserting table of contents at cursor position. If the table of contents already exists at cursor position, you can customize its properties.

CSHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableTableOfContentsDialog(true).EnableSelection(true).EnableSfdtExport(tru
e).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open TableOfContents Dialog
            documenteditor.showDialog('TableOfContents');
        });
    });
</script>
```

TABLE-OF-CONTENTS-DIALOG.CS

--

Styles Dialog

This dialog allows managing the styles in a document. It will display all the styles in the document with options to modify the properties of the existing style or create new style with the help of 'Style dialog'.

CHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableStyleDialog(true).EnableStylesDialog(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open Styles Dialog
            documenteditor.showDialog('Styles');
        });
    });
</script>
```

STYLES-DIALOG.CS

--

Style dialog

You can directly use this dialog for modifying any existing style or add new style by providing the style name.

CHTML

```
@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableStyleDialog(true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
```

```

    var button = document.getElementById('dialog');
    button.addEventListener('click', function () {
        // To open Style Dialog
        documenteditor.showDialog('Style');
    });
});
</script>

```

STYLE-DIALOG.CS

List dialog

This dialog allows creating a new list or modifying existing lists in the document.

CSHTML

```

@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableListDialog(true).EnableSelection(true).EnableSfdtExport(true).Render(
)
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open List Dialog
            documenteditor.showDialog('List');
        });
    });
</script>

```

LIST-DIALOG.CS

Borders and shading dialog

This dialog allows customizing the border style, border width, and background color of the table or selected cells.

CSHTML

```

@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableBordersAndShadingDialog(true).EnableSelection(true).EnableSfdtExport(
true).Render()

```

```

</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open BordersAndShading Dialog
            documenteditor.showDialog('BordersAndShading');
        });
    });
</script>

```

BORDERS-AND-SHADING-DIALOG.CS

Table options dialog

This dialog allows customizing the default cell margins and spacing between each cells of the selected table.

CSHTML

```

@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableTableOptionsDialog(true).EnableSelection(true).EnableSfdtExport(true)
.Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open TableOptions Dialog
            documenteditor.showDialog('TableOptions');
        });
    });
</script>

```

TABLE-OPTIONS-DIALOG.CS

Table properties dialog

This dialog allows customizing the table, row, and cell properties of the selected table.

CSHTML

```

@Html.EJS().Button("dialog").Content("Dialog").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableTablePropertiesDialog(true).EnableSelection(true).EnableSfdtExport(tr
ue).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open TableProperties Dialog
            documenteditor.showDialog('TableProperties');
        });
    });
</script>

```

TABLE-PROPERTIES-DIALOG.CS

Page setup dialog

This dialog allows customizing margins, size, and layout options for pages of the section.

CSHTML

```

@Html.EJS().Button("dialog").Content("Dialog").Render()
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnablePageSetupDialog(true).EnableSelection(true).EnableSfdtExport(true).Re
nder()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        var button = document.getElementById('dialog');
        button.addEventListener('click', function () {
            // To open PageSetup Dialog
            documenteditor.showDialog('PageSetup');
        });
    });
</script>

```

PAGE-SETUP-DIALOG.CS

See Also

- [Feature module](#)

Chart

Document Editor provides chart preservation support. Using Document Editor, you can see the chart reports from your Word document.

CSHTML

```
@Html.EJS().DocumentEditor("container").Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor = documenteditorElement.ej2_instances[0];
        documenteditor.resize();
        var sfdt =
        {
            "sections": [
                {
                    "sectionFormat": {
                        "pageWidth": 612, "pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72, "bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages": false, "headerDistance": 36, "footerDistance": 36, "bidi": false,
                    },
                    "blocks": [
                        {
                            "paragraphFormat": {
                                "textAlignment": "Center", "afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal",
                            },
                            "listFormat": {},
                            "characterFormat": {
                                "bold": true, "fontSize": 12, "fontFamily": "Verdana", "fontSizeBidi": 12, "fontFamilyBidi": "Verdana",
                            },
                            "inlines": [
                                {
                                    "characterFormat": {
                                        "bold": true, "fontSize": 14, "fontFamily": "Verdana", "fontColor": "#17365DFF", "styleName": "a", "fontSizeBidi": 14, "fontFamilyBidi": "Verdana",
                                    },
                                    "text": "Northwind Management Report"
                                },
                                {
                                    "paragraphFormat": {
                                        "afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {},
                                    },
                                    "characterFormat": {
                                        "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana",
                                    },
                                    "inlines": [
                                        {
                                            "paragraphFormat": {
                                                "afterSpacing": 0, "styleName": "Normal", "listFormat": {},
                                            },
                                            "characterFormat": {
                                                "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana",
                                            },
                                            "text": "This management report provides information obtained through data analysis, regarding the performance of Northwind Traders. This report will pay particular attention to the best-selling products of our company."
                                        },
                                        {
                                            "characterFormat": {
                                                "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana",
                                            },
                                            "text": "The best-selling products of Northwind Traders"
                                        },
                                        {
                                            "characterFormat": {
                                                "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman",
                                            },
                                            "text": "Company as follows:"
                                        }
                                    ],
                                },
                                {
                                    "paragraphFormat": {
                                        "afterSpacing": 0, "styleName": "Normal", "listFormat": {},
                                    },
                                    "characterFormat": {
                                        "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman",
                                    },
                                    "inlines": [
                                        {
                                            "characterFormat": {
                                                "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman",
                                            },
                                            "text": "S.No"
                                        }
                                    ],
                                },
                                {
                                    "cellFormat": {
                                        "borders": {
                                            "top": {
                                                "color": "#4472C4FF"
                                            }
                                        }
                                    }
                                }
                            ]
                        }
                    ]
                }
            ]
        }
```

```

", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "
space": 0}, "left": {"color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Sing
le", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "bottom": {"color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Singl
e", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000
", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "spa
ce": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "Non
e", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "
hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space":
0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "li
neWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#4472C4F
F", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth":
13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422
465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"columnIndex": 0}, {
"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "charact
erFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verd
ana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Product
Name"}]}], "cellFormat": {"borders": {"top": {"color": "#4472C4FF", "hasNoneStyle"
: false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left"
: {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth":
0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": fa
lse, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom":
{"color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0
.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle
": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonal
Up": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0
, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": fa
lse, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"
color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shad
ow": false, "space": 0}}, "shading": {"backgroundColor": "#4472C4FF", "foregroundColo
r": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.8600006103515
6, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan"
: 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"columnIndex": 1}, {"blocks": [{"para
graphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "in
lines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBi
di": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Sum of Sales(in
$)"}]}], "cellFormat": {"borders": {"top": {"color": "#4472C4FF", "hasNoneStyle": f
alse, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {
"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.
5, "shadow": false, "space": 0}, "right": {"color": "#4472C4FF", "hasNoneStyle": fals
e, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"
color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5
, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle":
false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp
": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "
shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": fals
e, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"co
lor": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow
": false, "space": 0}}, "shading": {"backgroundColor": "#4472C4FF", "foregroundColo
r": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 37.720001220703125
, "preferredWidthType": "Percent", "cellWidth": 117.95841899993776, "columnSpan":
1, "rowSpan": 1, "verticalAlignment": "Top"}, {"columnIndex": 2}], "rowFormat": {"hei
ght": 14.399999618530273, "allowBreakAcrossPages": true, "heightType": "Exactly",
"isHeader": false, "borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false,
"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"colo
r": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "sh

```



```

adow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point"},"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"1"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Côte de Blaye"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"141.396"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1}]}]

```

```
space":0}, "bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "horizontal":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "vertical":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}}, "shading":{"backgroundColor":"#D9E2F3FF", "foregroundColor":"empty", "textureStyle":"TextureNone"}, "preferredWidth":37.720001220703125, "preferredWidthType":"Percent", "cellWidth":117.95841899993776, "columnSpan":1, "rowSpan":1, "verticalAlignment":"Top", "columnIndex":2}], "rowFormat":{"height":14.399999618530273, "allowBreakAcrossPages":true, "heightType":"Exactly", "isHeader":false, "borders":{"top":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "left":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "horizontal":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "vertical":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}}, "gridBefore":0, "gridBeforeWidth":0, "gridAfterWidth":0, "gridAfterWidthType":"Point", "cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal", "listFormat":{}}, "characterFormat":{}, "inlines":[{"characterFormat":{"fontSize":10, "fontFamily":"Verdana", "fontSizeBidi":10, "fontFamilyBidi":"Times New Roman"}, "text":"2"}]}], "cellFormat":{"borders":{"top":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "left":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "horizontal":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "vertical":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}}, "shading":{"backgroundColor":"#FFFFFF", "foregroundColor":"empty", "textureStyle":"TextureNone"}, "preferredWidth":13.420000076293945, "preferredWidthType":"Percent", "cellWidth":64.71214527422465, "columnSpan":1, "rowSpan":1, "verticalAlignment":"Top", "columnIndex":0}, {"blocks":[{"paragraphFormat":{"styleName":"Normal", "listFormat":{}}, "characterFormat":{}, "inlines":[{"characterFormat":{"fontSize":10, "fontFamily":"Verdana", "fontSizeBidi":10, "fontFamilyBidi":"Times New Roman"}, "text":"Thüringer Rostbratwurst"}]}], "cellFormat":{"borders":{"top":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "left":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, "diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "horizontal":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, "vertical":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}}, "shading":{"backgroundColor":"#FFFFFF", "foregroundColor":"empty", "textureStyle":"TextureNone"}, "preferredWidth":13.420000076293945, "preferredWidthType":"Percent", "cellWidth":64.71214527422465, "columnSpan":1, "rowSpan":1, "verticalAlignment":"Top", "columnIndex":0}
```

```

"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"80.368"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],
"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"rightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{
"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"3"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},

```

```

"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Raclette Courdavault"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"71.155"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.9584189993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],
"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0}

```

```

th":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAf
terWidthType":"Point"}},{ "cells": [{"blocks": [{"paragraphFormat": {"styleName
": "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat
": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "
Times New
Roman"}, "text": "4"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "has
NoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space"
: 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "has
NoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0},
"vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineW
idth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF",
"foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.
420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465
, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {""bl
ocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}, "characterF
ormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana
", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Tarte au
sucre
"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": fal
se, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"c
olor": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5,
"shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false,
"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"co
lor": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "
shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": fa
lse, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp":
{"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "sh
adow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false,
"lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"colo
r": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow":
false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor":
"empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "p
REFERREDWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "
rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 1}, {""blocks": [{"paragrap
hFormat": {"styleName": "Normal", "listFormat": {}, "characterFormat": {}, "inline
s": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi":
10, "fontFamilyBidi": "Times New
Roman"}, "text": "47.234"}]}, {"characterFormat": {}, "bookmarkType": 1, "name": "_GoB
ack"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle":
false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left":
{"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0
.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": fal
se, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {
"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.
5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle"
: false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalU
p": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0,
"shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": fal
se, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"c
olor": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shado

```

```

w":false,"space":0}},{"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}]],"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},{"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"},"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"bookmarkType":0,"name":"_GoBack"},{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"5"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Camembert Pierrot"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"p

```



```

referredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"
rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks": [{ "paragraph
hFormat": { "styleName": "Normal", "listFormat": {}, "characterFormat": {}, "inlines
": [{ "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi":
10, "fontFamilyBidi": "Times New
Roman"}, "text": "46.825"} ]}], "cellFormat": { "borders": { "top": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false
, "space": 0}, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Si
ngle", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "s
pace": 0}, "bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sin
gle", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": { "color": "#000
000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "s
pace": 0}, "diagonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "N
one", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": { "color": "#000000"
, "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space
": 0}, "vertical": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "
lineWidth": 0, "shadow": false, "space": 0}}, "shading": { "backgroundColor": "#D9E2F
3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth
": 37.720001220703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899
993776, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2
}], "rowFormat": { "height": 14.399999618530273, "allowBreakAcrossPages": true, "he
ightType": "Exactly", "isHeader": false, "borders": { "top": { "color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0}, "right": { "color": "#8EADBFF", "has
NoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0}, "bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": { "color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space"
: 0}, "diagonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": { "color": "#8EADBFF", "h
asNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spac
e": 0}, "vertical": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sing
le", "lineWidth": 0.5, "shadow": false, "space": 0}}, "gridBefore": 0, "gridBeforeWid
th": 0, "gridBeforeWidthType": "Point", "gridAfter": 0, "gridAfterWidth": 0, "gridAf
terWidthType": "Point"} }, { "cells": [{ "blocks": [{ "paragraphFormat": { "styleName"
: "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{ "characterFormat
": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "
Times New
Roman"}, "text": "6"} ]}], "cellFormat": { "borders": { "top": { "color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0}, "right": { "color": "#8EADBFF", "has
NoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0}, "bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": { "color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space"
: 0}, "diagonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": { "color": "#000000", "has
NoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0},
"vertical": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineW
idth": 0, "shadow": false, "space": 0}}, "shading": { "backgroundColor": "#FFFFFFF",
"foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.
420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465
, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, { "bl
ocks": [{ "paragraphFormat": { "styleName": "Normal", "listFormat": {}, "characterF

```

```

ormat":{}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Gnocchi di nonna Alice"}]], "cellFormat": {"borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 1}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "42.593"}]]}, {"cellFormat": {"borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 37.720001220703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899993776, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2}], "rowFormat": {"height": 14.3999999618530273, "allowBreakAcrossPages": true, "heightType": "Exactly", "isHeader": false, "borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "vertical": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}}, "gridBefore": 0, "gridBeforeWidth": 0, "gridBeforeWidthType": "Point", "gridAfter": 0, "gridAfterWidth": 0, "gridAfterWidthType": "Point"}, {"cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New

```



```

Roman","text":"7"]]]], "cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Manjimup Dried Apples"]}]}, "cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"41.819"]}]}, "cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.9584189993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2
}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","

```

```

hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},
"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point",
"gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"},
{"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"}},
"text":"8"}]}],
"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},
"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},
"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,
"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},
{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"}},
"text":"Alice Mutton"}]}],
"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},
"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},
"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},
{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"}},
"text":"32.698"}]}],
"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false

```

```
"space":0}, {"color":"#8EADBFFF","hasNoneStyle":false,"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"horizontal":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"vertical":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}}, {"shading":{"backgroundColor":"#FFFFFF", "foregroundColor":"empty", "textureStyle":"TextureNone"}}, {"preferredWidth":37.720001220703125, "preferredWidthType":"Percent", "cellWidth":117.95841899993776, "columnSpan":1, "rowSpan":1, "verticalAlignment":"Top"}, {"columnIndex":2}], {"rowFormat":{"height":14.399999618530273, "allowBreakAcrossPages":true, "heightType":"Exactly", "isHeader":false, "borders":{"top":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"left":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"horizontal":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"vertical":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}}}, {"gridBefore":0, "gridBeforeWidth":0, "gridBeforeWidthType":"Point", "gridAfter":0, "gridAfterWidth":0, "gridAfterWidthType":"Point"}}, {"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal", "listFormat":{}}, {"characterFormat":{}}, {"inlines":[{"characterFormat":{"fontSize":10, "fontFamily":"Verdana", "fontSizeBidi":10, "fontFamilyBidi":"Times New Roman"}}, {"text":"9"}]}]}, {"cellFormat":{"borders":{"top":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"left":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"horizontal":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"vertical":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}}}, {"shading":{"backgroundColor":"#D9E2F3FF", "foregroundColor":"empty", "textureStyle":"TextureNone"}}, {"preferredWidth":13.420000076293945, "preferredWidthType":"Percent", "cellWidth":64.71214527422465, "columnSpan":1, "rowSpan":1, "verticalAlignment":"Top"}, {"columnIndex":0}, {"blocks":[{"paragraphFormat":{"styleName":"Normal", "listFormat":{}}, {"characterFormat":{}}, {"inlines":[{"characterFormat":{"fontSize":10, "fontFamily":"Verdana", "fontSizeBidi":10, "fontFamilyBidi":"Times New Roman"}}, {"text":"Carnarvon Tigers"}]}]}, {"cellFormat":{"borders":{"top":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"left":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"right":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"bottom":{"color":"#8EADBFFF", "hasNoneStyle":false, "lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0}, {"diagonalDown":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"diagonalUp":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"horizontal":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}, {"vertical":{"color":"#000000", "hasNoneStyle":false, "lineStyle":"None", "lineWidth":0, "shadow":false, "space":0}}}, {"shading":{"backgroundColor":"#D9E2F3FF", "foregroundColor":"empty", "textureStyle":"TextureNone"}}, {"preferredWidth":13.420000076293945, "preferredWidthType":"Percent", "cellWidth":64.71214527422465, "columnSpan":1, "rowSpan":1, "verticalAlignment":"Top"}, {"columnIndex":0}]]], [{"text":"Carnarvon Tigers"}]}
```

```

":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"29.171"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}]},"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"rightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"10"}]}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}}

```

```

":0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0},
"vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Rössle Sauerkraut."}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 1}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "25.696"}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 37.720001220703125, "preferredWidthType": "Percent", "cellWidth": 117.9584189993776, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2}], "rowFormat": {"height": 14.399999618530273, "allowBreakAcrossPages": true, "rightType": "Exactly", "isHeader": false, "borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EADBFF", "h

```

```

NoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}]],"grid":[64.71214527422465,292.87942351880633,117.95841899993776],"tableFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"cellSpacing":0,"leftIndent":0,"tableAlignment":"Left","topMargin":0,"rightMargin":0.5,"leftMargin":0.5,"bottomMargin":0,"preferredWidth":475.54998779296875,"preferredWidthType":"Point","bidi":false,"allowAutoFit":true},"description":null,"title":null},{ "paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{"fontFamily":"Calibri","fontColor":"#000000FF","fontFamilyBidi":"Calibri"},"inlines":[]},{ "paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{"inlines":{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"The best-selling product of the company is Cote de Blaye, being part of the Beverages"}},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"category. The contribution of this product to the sum of our sales is $141.396."}}},{ "paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]},{ "paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]},{ "paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"chartLegend":{"position":"Right","chartTitleArea":{"fontName":"+mn-1t","fontSize":9,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"foreColor":"000000","rgb":"#000000"},"line":{"color":"808080","rgb":"#808080"}}}},"chartTitleArea":{"fontName":"+mn-1t","fontSize":14,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"foreColor":"000000","rgb":"#000000"},"line":{"color":"000000","rgb":"#000000"}}},"chartArea":{"foreColor":"#FFFFFF","plotArea":{"foreColor":"#000000FF"},"chartCategory":[{"chartData":[{"yValue":141.396}],"categoryXName":"Côte de Blaye"}, {"chartData":[{"yValue":80.368}],"categoryXName":"Thüringer Rostbratwurst"}, {"chartData":[{"yValue":71.155}],"categoryXName":"Raclette Courdavault"}, {"chartData":[{"yValue":47.234}],"categoryXName":"Tarte au sucre"}, {"chartData":[{"yValue":46.825}],"categoryXName":"Camembert Pierrot"}, {"chartData":[{"yValue":42.593}],"categoryXName":"Gnocchi di nonna Alice"}, {"chartData":[{"yValue":41.819}],"categoryXName":"Manjimup Dried Apples"}, {"chartData":[{"yValue":32.698}],"categoryXName":"Alice

```



```

Mutton"}, {"chartData": [{"yValue": 29.171}], "categoryXName": "Carnarvon
Tigers"}, {"chartData": [{"yValue": 25.696}], "categoryXName": "Rössle
Sauerkraut"}], "chartSeries": [{"dataPoints": [{"fill": {"foreColor": "4472c4", "r
gb": "#4472c4"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColo
r": "ed7d31", "rgb": "#ed7d31"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fi
ll": {"foreColor": "a5a5a5", "rgb": "#a5a5a5"}, "line": {"color": "ffffff", "rgb": "#
ffffff"}}, {"fill": {"foreColor": "ffc000", "rgb": "#ffc000"}, "line": {"color": "ff
ffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "5b9bd5", "rgb": "#5b9bd5"}, "line
": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "70ad47", "rgb": "#
70ad47"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "26
4379", "rgb": "#264379"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"
foreColor": "9f480e", "rgb": "#9f480e"}, "line": {"color": "ffffff", "rgb": "#ffffff
"}}, {"fill": {"foreColor": "636363", "rgb": "#636363"}, "line": {"color": "ffffff",
"rgb": "#ffffff"}}, {"fill": {"foreColor": "9a7200", "rgb": "#9a7200"}, "line": {"co
lor": "ffffff", "rgb": "#ffffff"}}], "seriesName": "Sales"}], "chartPrimaryCategor
yAxis": {"chartTitle": null, "chartTitleArea": {"layout": {}, "dataFormat": {"fill"
: {}, "line": {}}, "categoryType": "Automatic", "fontSize": 11, "fontName": "Calibri
", "numberFormat": "General", "maximumValue": 0, "minimumValue": 0, "majorUnit": 0, "
hasMajorGridLines": false, "hasMinorGridLines": false, "majorTickMark": "TickMark
_Outside", "minorTickMark": "TickMark_None", "tickLabelPosition": "TickLabelPosi
tion_NextToAxis"}, "chartPrimaryValueAxis": {"chartTitle": null, "chartTitleArea
": {"layout": {}, "dataFormat": {"fill": {}, "line": {}}, "fontSize": 11, "fontName":
"Calibri", "maximumValue": 0, "minimumValue": 0, "majorUnit": 0, "hasMajorGridLines
": false, "hasMinorGridLines": false, "majorTickMark": "TickMark_Outside", "minorT
ickMark": "TickMark_None", "tickLabelPosition": "TickLabelPosition_NextToAxis"}
, "chartTitle": "Best Selling
Products", "chartType": "Pie", "gapWidth": 0, "overlap": 0, "height": 225, "width": 43
2}}], {"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFor
mat": {}, "inlines": []}, {"paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1, "
lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}}, "character
Format": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyB
idi": "Verdana"}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "V
erdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text"
: "According to the above chart, the total count of the selling products is
24 and the average
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "sales attributed to
this product is $ 5.891 with highest sale $ 15.810 in the month of May in
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "2014. In the same
year, in the month of March the same product reached the amount of $
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "15.019. These were the
highest sales of the product among the other products for the year
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "2014."}]]], "headersFoo
ters": {}], "characterFormat": {"bold": false, "italic": false, "fontSize": 11, "fon
tFamily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignm
ent": "Normal", "highlightColor": "NoColor", "fontColor": "#000000", "fontSizeBidi
": 11, "fontFamilyBidi": "Calibri"}, "paragraphFormat": {"leftIndent": 0, "rightInd
ent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSp
acing": 8, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "list
Format": {}, "bidi": false}, "defaultTabWidth": 36, "enforcement": false, "hashValue
": "", "saltValue": "", "formatting": false, "protectionType": "NoProtection", "styl
es": [{"name": "Normal", "type": "Paragraph", "paragraphFormat": {"listFormat": {}
}, "characterFormat": {}, "next": "Normal"}, {"name": "Heading
1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 3

```

```

,"lineSpacing":1,"lineSpacingType":"Multiple","outlineLevel":"Level1","listFormat":{}}, {"characterFormat":{"bold":true,"fontSize":16,"fontFamily":"Arial","boldBidi":true,"fontSizeBidi":16,"fontFamilyBidi":"Arial"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"}, {"name":"Heading 1 Char","type":"Character","characterFormat":{"bold":true,"fontSize":16,"fontFamily":"Arial","boldBidi":true,"fontSizeBidi":16,"fontFamilyBidi":"Arial"},"basedOn":"Default Paragraph Font"}, {"name":"Default Paragraph Font","type":"Character","characterFormat":{}}, {"name":"Balloon Text","type":"Paragraph","paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","listFormat":{}}, {"characterFormat":{"fontSize":9,"fontFamily":"Segoe UI","fontSizeBidi":9,"fontFamilyBidi":"Segoe UI"},"basedOn":"Normal","link":"Balloon Text Char"}, {"name":"Balloon Text Char","type":"Character","characterFormat":{"fontSize":9,"fontFamily":"Segoe UI","fontSizeBidi":9,"fontFamilyBidi":"Segoe UI"},"basedOn":"Default Paragraph Font"}, {"name":"a","type":"Character","characterFormat":{},"basedOn":"Default Paragraph Font"}, {"name":"Heading 2","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level2","listFormat":{}}, {"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"}, {"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"}, {"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level3","listFormat":{}}, {"characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3 Char","next":"Normal"}, {"name":"Heading 3 Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"}, {"name":"Heading 4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level4","listFormat":{}}, {"characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4 Char","next":"Normal"}, {"name":"Heading 4 Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"}, {"name":"Heading 5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level5","listFormat":{}}, {"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 5 Char","next":"Normal"}, {"name":"Heading 5 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"}, {"name":"Heading 6","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level6","listFormat":{}}, {"characterFormat":{"fontFamily":"Calibri

```



```
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6  
Char", "next": "Normal"}, {"name": "Heading 6  
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri  
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph  
Font"}], "lists": [], "abstractLists": []};  
    // preserve chart  
    documenteditor.open(JSON.stringify(sfdt));  
  });  
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Supported Chart Types

The following chart types are supported in document editor

- Scatter_Markers
- Bubble
- Area
- Area_Stacked
- AreaStacked100
- Bar_Clustered
- Bar_Stacked
- BarStacked100
- Column_Clustered
- Column_Stacked
- ColumnStacked100
- Pie
- Doughnut
- Line
- Line_Markers
- LineMarkersStacked
- LineMarkersStacked_100
- Line_Stacked
- LineStacked100

Restrict Editing in Document Editor Component

Document Editor provides support to restrict editing. When the protected document includes range permission, then unique user or user group is only authorized to edit separate text area.

Set current user

You can use the `currentUser` property to authorize the current document user by name, email, or user group name.

The following code shows how to set `currentUser`

```
`typescript
```

```
container.documentEditor.currentUser = 'engineer@mycompany.com';
```

```
,
```

Highlighting the text area

You can highlight the editable region of the current user using the `userColor` property.

The following code shows how to set `userColor`.

```
`typescript
```

```
container.documentEditor.userColor = '#fff000';
```

```
,
```

Restrict Editing Pane

Restrict Editing Pane provides the following options to manage the document:

- To apply formatting restrictions to the current document, select the allow formatting check box.
- To apply editing restrictions to the current document, select the read only check box.
- To add users to the current document, select more users option and add user from the popup dialog.
- To include range permission to the current document, select parts of the document and choose users who are allowed to freely edit them from the listed check box.
- To apply the chosen editing restrictions, click the **YES, START ENFORCING PROTECTION** button. A dialog box displays asking for a password to protect.
- To stop protection, select **STOP PROTECTION** button. A dialog box displays asking for a password to stop protection.
- [How to protect the document in form filling mode](#)
- [How to protect the document in comments only mode](#)
- [How to protect the document in track changes only mode](#)

Spell Check in Document Editor Component

Document editor supports performing spell checking for any input text. You can perform spell checking for the text in Document Editor and it will provide suggestions for the mis-spelled words through dialog and in context menu.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableSpellChecker(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documenteditor.spellChecker.languageID = 1033 //LCID of "en-us";
        documenteditor.spellChecker.removeUnderline = false;
        documenteditor.spellChecker.allowSpellCheckAndSuggestion = true;
    });
</script>
```

SPELL-CHECKER.CS

Features

- Supports context menu suggestions.
- Provides build-in options to Ignore, Ignore All, Change, Change All for error words in spell checker dialog.

Enable SpellCheck

To enable spell check in DocumentEditor, set `enableSpellCheck` property as `true` and then configure `SpellCheckSettings`.

Disable SpellCheck

To disable spell check in DocumentEditor, set `enableSpellCheck` property as `false` or remove `enableSpellCheck` property initialization code. The default value of this property is false.

Spell check settings

Remove Underline

By default, mis-spelled words are marked with squiggly line. You can also disable this behavior by enabling the `removeUnderline` API and now, the squiggly lines will never be rendered for mis-spelled words.

```
`typescript
```

```
this.container.documentEditor.spellChecker.removeUnderline = false;
```

```
,
```

AllowSpellCheckAndSuggestion

By default, on performing spell check in Document Editor, both spelling and suggestions of the mis-spelled words will be retrieved, and this mis-spelled words can be corrected through context menu suggestions. You can modify this behavior using the `allowSpellCheckAndSuggestion` API, which will perform only spell check.

```
`typescript
```

```
this.container.documentEditor.spellChecker.allowSpellCheckAndSuggestion = false;
```

```
,
```

LanguageID

Document Editor provides multi-language spell check support. You can add as many languages (dictionaries) in the server-side and to use that language for spell checking in Document Editor, it must be matched with `languageID` you pass in the Document Editor.

```
`typescript
```

```
this.container.documentEditor.spellChecker.languageID = 1033; //LCID of "en-us";
```

```
,
```

EnableOptimizedSpellCheck

Document editor provides option to spellcheck page by page when loading the documents. The default value of this property is false, so when opening the document spellcheck web API will be called for each word in the document. To optimize the frequency of spellcheck web API calls, you can enable this property.

```
`typescript
```

```
this.container.documentEditor.spellChecker.enableOptimizedSpellCheck = true;
```

```
,
```

Spell check dictionary cache

Starting from **v20.1.0.xx**, the performance and memory usage of spell checker has been optimized by adding a static method to initialize the dictionaries with specified cache count.

By default, the spell checker holds only one language dictionary in memory. If you want to hold multiple dictionaries in memory, you need to set the cache limit by using **InitializeDictionaries** method as in the below example.

```
`csharp
```

```
List<DictionaryData> spellDictCollection = new List<DictionaryData>();
```

```
string personalDictPath = string.Empty;
```

```
int cacheCount = 2;
```

```
// Initialize dictionaries
```

```
SpellChecker.InitializeDictionaries(spellDictCollection, personalDictPath, cacheCount);
```

```
,
```

If dictionaries are initialized using **InitializeDictionaries** method, then default constructor of the **SpellChecker** should be used to check spelling and get suggestion as in the below example code, it will prevent reinitialization of already loaded dictionaries.

```
`csharp
```

```
public string SpellCheck([FromBody] SpellCheckJsonData spellChecker)
```

```
{
```

```
try
```

```
{
```

```
SpellChecker spellCheck = new SpellChecker();
```

```
spellCheck.GetSuggestions(spellChecker.LanguageID, spellChecker.TexttoCheck,
```

```
spellChecker.CheckSpelling, spellChecker.CheckSuggestion, spellChecker.AddWord);
```

```
return Newtonsoft.Json.JsonConvert.SerializeObject(spellCheck);
```

```
}
```

```
catch
```

```
{
```

```
return "{\"SpellCollection\": [], \"HasSpellingError\": false, \"Suggestions\": null}";
}
}
,
```

Previously on every `SpellChecker.GetSuggestion()` method call, the `.aff` and dictionary data will be parsed to generate suggestion for miss spelled word. But, starting from `v20.1.0.xx`, the `.aff` and dictionary data will be parsed only for the first time alone while calling `SpellChecker.GetSuggestion()` method.

Add new root word and possible words to dictionary

If you find any root word is missing in the dictionary file, then you can add that new root word and the rule to form the possible words to dictionary file using `AddNewWord` API in the server-side Spell check library.

Note: 1. The rules are framed automatically using the root word, the possible words and affix file.

2. If you pass null for the parameters `affPath` and `possibleWords`, then it will add a single root word to dictionary.

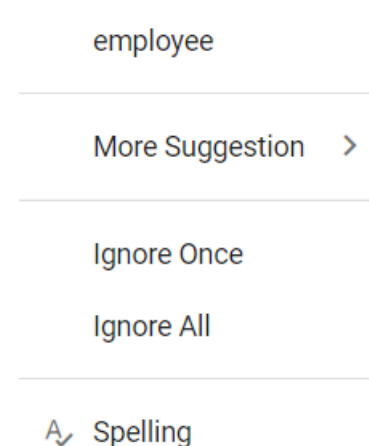
3. This API is included starting from `v20.2.0.xx`.

The following code example demonstrates how to add a new root word to the dictionary along with the rule to form the possible words.

```
`csharp
SpellChecker spellChecker = new SpellChecker();
// Adds the specified new root word to the dictionary along with the rule to form the possible words.
spellChecker.AddNewWord("en.dic", "en.aff", "construct", new string[] { "constructs", "reconstruct",
"constructed", "constructive" });
,
```

Context menu

Right click on error word to open the context menu with spell check options.



Suggestions

Context menu shows the suggestions for mis-spelled words. By clicking on the required word from suggestion, the error word gets replaced automatically.

Add To Dictionary

Using this option, you can add the current word to the dictionary. So that the spell checker does not consider that word as error in future.

Ignore Once and Ignore All

If you do not wish to add the word to dictionary and do not want to show error, use Ignore Once or Ignore All options.

Ignore: ignore only the current occurrence of a word from error.

Ignore All: ignore all occurrence of a word from error in the entire document.

Spelling

Using this option, you can open spell check dialog.

Spelling Editor

×

Spelling

wrd

IGNORE

IGNORE ALL

ADD TO DICTIONARY

Suggestions

ward

word

rd

wad

CHANGE

CHANGE ALL

CANCEL

- Refer to the [Spell checker](#) link for configuring spell checker in server-side.

Globalization

Localization

The [Localization](#) library allows to localize default text content of the DocumentEditor. The document editor component has static text on some features (like find & replace, context-menu, dialogs) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the locale value and translation object. Refer the sample link [RTL](#).

Note: Refer the [Locale](#).

Document Editor

The following list of properties and its values are used in the document editor.

Locale keywords | Text

New | New

Open | Open

Undo | Undo

Redo | Redo

Image | Image

Table | Table

Link | Link

Bookmark | Bookmark

Table of Contents | Table of Contents

HEADING - - - - 1 | HEADING - - - - 1

HEADING - - - - 2 | HEADING - - - - 2

HEADING - - - - 3 | HEADING - - - - 3

Header | Header

Footer | Footer

Page Setup | Page Setup

Page Number | Page Number

Break | Break

Find | Find

Local Clipboard | Local Clipboard

Restrict Editing | Restrict Editing

Upload from computer | Upload from computer

By URL | By URL

Page Break | Page Break

Section Break | Section Break

Header And Footer | Header & Footer

Options | Options

Levels | Levels

Different First Page | Different First Page

Different header and footer for odd and even pages | Different header and footer for odd and even pages.

Different Odd And Even Pages | Different Odd & Even Pages

Different header and footer for first page | Different header and footer for first page.

Position | Position

Header from Top | Header from Top

Footer from Bottom | Footer from Bottom

Distance from top of the page to top of the header | Distance from top of the page to top of the header.

Distance from bottom of the page to bottom of the footer | Distance from bottom of the page to bottom of the footer.

Aspect ratio | Aspect ratio

W | W

H | H

Width | Width

Height | Height

Text | Text

Paragraph | Paragraph

Fill | Fill

Fill color | Fill color

Border Style | Border Style

Outside borders | Outside borders

All borders | All borders

Inside borders | Inside borders

Left border | Left border

Inside vertical border | Inside vertical border

Right border | Right border

Top border | Top border

Inside horizontal border | Inside horizontal border

Bottom border | Bottom border

Border color | Border color

Border width | Border width

Cell | Cell

Merge cells | Merge cells

Insert Or Delete | Insert / Delete

Insert columns to the left | Insert columns to the left

Insert columns to the right | Insert columns to the right

Insert rows above | Insert rows above

Insert rows below | Insert rows below

Delete rows | Delete rows

Delete columns | Delete columns

Cell Margin | Cell Margin

Top | Top

Bottom | Bottom

Left | Left

Right | Right

Align Text | Align Text

Align top | Align top

Align bottom | Align bottom

Align center | Align center

Number of heading or outline levels to be shown in table of contents | Number of heading or outline levels to be shown in table of contents.

Show page numbers | Show page numbers

Show page numbers in table of contents | Show page numbers in table of contents.

Right align page numbers | Right align page numbers

Right align page numbers in table of contents | Right align page numbers in table of contents.

Use hyperlinks | Use hyperlinks

Use hyperlinks instead of page numbers | Use hyperlinks instead of page numbers.

Font | Font

Font Size | Font Size

Font color | Font color

Text highlight color | Text highlight color

Clear all formatting | Clear all formatting

Bold Tooltip | Bold (Ctrl+B)

Italic Tooltip | Italic (Ctrl+I)

Underline Tooltip | Underline (Ctrl+U)

Strikethrough | Strikethrough

Superscript Tooltip | Superscript (Ctrl+Shift++)

Subscript Tooltip | Subscript (Ctrl+=)

Align left Tooltip | Align left (Ctrl+L)

Center Tooltip | Center (Ctrl+E)

Align right Tooltip | Align right (Ctrl+R)

Justify Tooltip | Justify (Ctrl+J)

Decrease indent | Decrease indent

Increase indent | Increase indent

Line spacing | Line spacing

Bullets | Bullets

Numbering | Numbering

Styles | Styles

Manage Styles | Manage Styles

Page | Page

of | of

Fit one page | Fit one page

Spell Check | Spell Check

Underline errors | Underline errors

Fit page width | Fit page width

Update | Update

Cancel | Cancel

Insert | Insert

No Border | No Border

Create a new document | Create a new document.

Open a document | Open a document.

Undo Tooltip | Undo the last operation (Ctrl+Z).

Redo Tooltip | Redo the last operation (Ctrl+Y).

Insert inline picture from a file | Insert inline picture from a file.

Insert a table into the document | Insert a table into the document

Create Hyperlink | Create a link in your document for quick access to web pages and files (Ctrl+K).

Insert a bookmark in a specific place in this document | Insert a bookmark in a specific place in this document.

Provide an overview of your document by adding a table of contents | Provide an overview of your document by adding a table of contents.

Add or edit the header | Add or edit the header.

Add or edit the footer | Add or edit the footer.

Open the page setup dialog | Open the page setup dialog.

Add page numbers | Add page numbers.

Find Text | Find text in the document (Ctrl+F).

Toggle between the internal clipboard and system clipboard | Toggle between the internal clipboard and system clipboard.

Access to system clipboard through script is denied due to browsers security policy. Instead, 1. You can enable internal clipboard to cut, copy and paste within the component. 2. You can use the keyboard shortcuts (Ctrl+X, Ctrl+C and Ctrl+V) to cut, copy and paste with system clipboard.

Current Page Number | The current page number is in the document. Click or tap to navigate specific page.

Read only | Read only

Protections | Protections

Error in establishing connection with web server | Error in establishing connection with web server

Single | Single

Double | Double

New comment | New comment

Comments | Comments

Print layout | Print layout

Web layout | Web layout

Text Form | Text Form

Check Box | Check Box

DropDown | Drop-Down

Update Fields | Update Fields

Update cross reference fields | Update cross reference fields

Hide properties pane | Hide properties pane

Show properties pane | Show properties pane

[Color Picker](#)

The following list of properties and its values are used in the color picker.

Locale keywords | Text

Apply | Apply

Cancel | Cancel

ModeSwitcher | Switch Mode

Insert footnote endnote

DocumentEditorContainer component provides support for inserting footnotes and endnotes through the in-built toolbar.



The footnotes and endnotes are both ways of adding extra bits of information to your writing outside of the main text. You can use footnotes and endnotes to add side comments to your work or to place other publications like books, articles, or websites.

Insert footnotes

Document editor exposes an API to insert footnotes at cursor position programmatically or can be inserted to the end of selected text.

CSHTML

```
@Html.EJS().Button("insertFootnote").Content("insert Footnote").Render()
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableEditor(
true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
    });
    document.getElementById('insertFootnote').addEventListener('click',
function () {
        documenteditor.editor.insertFootnote();
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Insert endnotes

Document editor exposes an API to insert endnotes at cursor position programmatically or can be inserted to the end of selected text.

CSHTML

```
@Html.EJS().Button("InsertEndnote").Content("Insert Endnote").Render()
<div id="documenteditor" style="width:100%;height:100%">

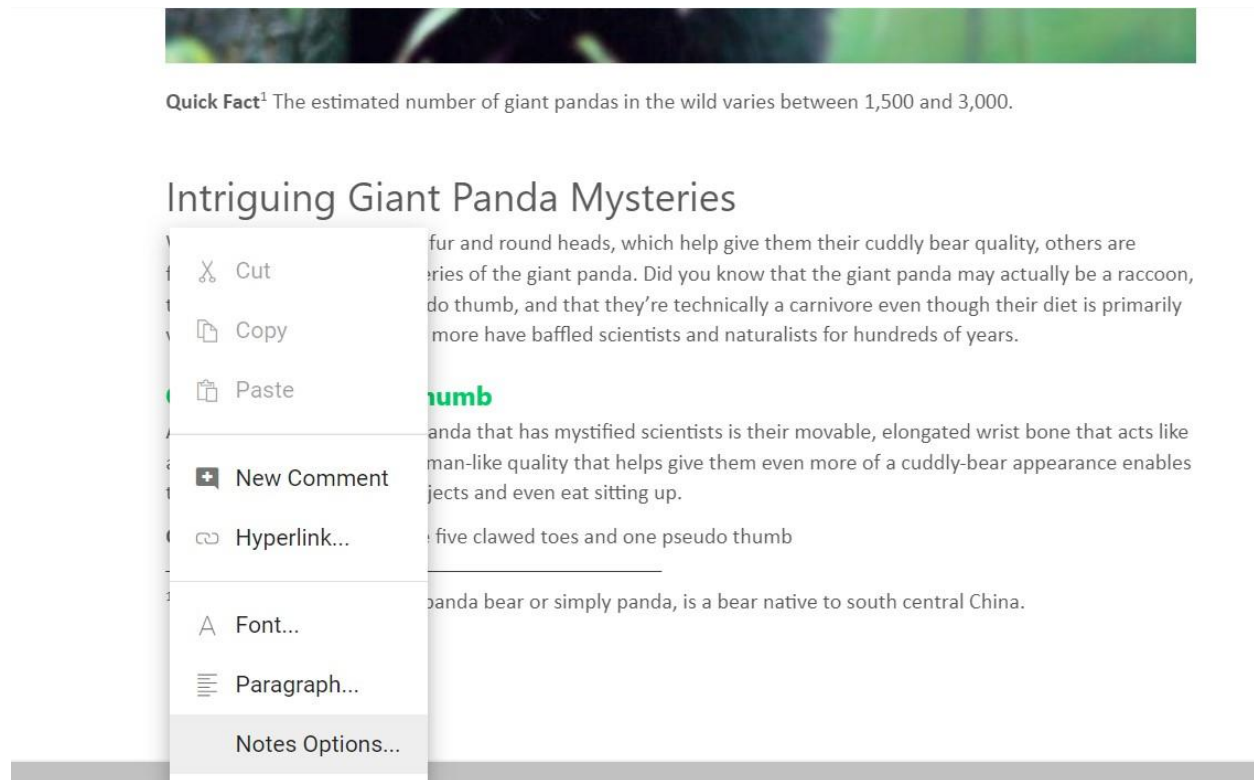
@Html.EJS().DocumentEditor("DocumentEditor").IsReadOnly(false).EnableEditor(
true).EnableSelection(true).EnableSfdtExport(true).Render()
</div>
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById('DocumentEditor').ej2_instances[0];
        documenteditor.resize();
    });
    document.getElementById('InsertEndnote').addEventListener('click',
function () {
        documenteditor.editor.insertEndnote();
    });
</script>
```

DOCUMENT-EDITOR.CS

```
public ActionResult Default()
{
    return View();
}
```

Update or edit footnotes and endnotes

You can update or edit the footnotes and endnotes using the built-in context menu shown up by right-clicking it. The footnote endnote dialog box popup and you can customize the number format and start at.



How To

How to override the keyboard shortcuts in document editor

Document editor triggers the [keyDown](#) event every time when any key is entered and provides an instance of `DocumentEditorKeyDownEventArgs`. You can use the `isHandled` property to override the keyboard shortcut behaviour.

Preventing default keyboard shortcut

The following code shows how to prevent the **CTRL + C** keyboard shortcut for copying selected content in document editor.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).Render()
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documentEditor.keyDown = function (args) {
            var keyCode = args.event.which || args.event.keyCode;
            var isCtrlKey = (args.event.ctrlKey || args.event.metaKey) ?
true : ((keyCode === 17) ? true : false);
            //67 is the character code for 'C'
            if (isCtrlKey && keyCode === 67) {
                //To prevent copy operation set isHandled to true
                args.isHandled = true;
            }
        }
    })
}
```

```
});
</script>
```

PREVENT-DEFAULT.CS

Override or define the keyboard shortcut

Override or define a new keyboard shortcut behaviour instead of preventing the keyboard shortcut.

For example, **Ctrl + S** keyboard shortcut saves the document in SFDT format by default, and there is no behaviour for **Ctrl + Alt + S**. The following code demonstrates how to override the **Ctrl + S** shortcut to save a document in DOCX format and define **Ctrl + Alt + S** to save the document in SFDT format.

CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnableSelection(true).EnableWordExport(true).enableSfdtExport(true).Render(
)
<script>
    var documenteditor;
    document.addEventListener('DOMContentLoaded', function () {
        documenteditor =
document.getElementById("container").ej2_instances[0];
        documentEditor.keyDown = function (args) {
            var keyCode = args.event.which || args.event.keyCode;
            var isCtrlKey = (args.event.ctrlKey || args.event.metaKey) ?
true : ((keyCode === 17) ? true : false);
            var isAltKey = args.event.altKey ? args.event.altKey : ((keyCode
=== 18) ? true : false);
            // 83 is the character code for 'S'
            if (isCtrlKey && !isAltKey && keyCode === 83) {
                //To prevent default save operation, set the isHandled
property to true
                args.isHandled = true;
                documentEditor.save('sample', 'Docx');
                args.event.preventDefault();
            } else if (isCtrlKey && isAltKey && keyCode === 83) {
                documentEditor.save('sample', 'Sfdt');
            }
        }
    });
</script>
```

OVERRIDE.CS

Context menu customization

How to customize context menu in Document Editor

Document Editor allows to add custom option in context menu. It can be achieved by using the `addCustomMenu()` method and custom action is defined using the `customContextMenuSelect`.

Add Custom Option

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // creating Custom Options
        let menuItems = [
            {
                text: 'Search In Google',
                id: 'search_in_google',
                iconCss: 'e-icons e-de-ctnr-find',
            },
        ];
        // adding Custom Options
        container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
        // custom Options Select Event
        container.documentEditor.customContextMenuSelect = function (args) {
            // custom Options Functionality
            let id = container.documentEditor.element.id;
            switch (args.id) {
                case id + 'search_in_google':
                    // To get the selected content as plain text
                    let searchContent =
                        container.documentEditor.selection.text;
                    if (
                        !container.documentEditor.selection.isEmpty &&
                        /\S/.test(searchContent)
                    ) {
                        window.open('http://google.com/search?q=' +
searchContent);
                    }
                    break;
            }
        };
    }
</script>
```

ADD-CUSTOM-MENU.CS

Customize custom option in context menu

Document Editor allows to customize the added custom option and also to hide or show default context menu.

Hide default context menu items

Using `addCustomMenu()` method, you can hide the default context menu, by setting second parameter as true.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // creating Custom Options
        let menuItems = [
            {
                text: 'Search In Google',
                id: 'search_in_google',
                iconCss: 'e-icons e-de-ctnr-find',
            },
        ];
        // The second parameter "true" hide the default context menu
        container.documentEditor.contextMenu.addCustomMenu(menuItems, true);
    }
</script>
```

HIDE-CONTEXT-MENU.CS

Customize added context menu items

The following code shows how to hide or show added custom option in context menu using the `customContextMenuBeforeOpen`.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // creating Custom Options
        let menuItems = [
            {
                text: 'Search In Google',
```

```

        id: 'search_in_google',
        iconCss: 'e-icons e-de-ctnr-find',
    },
];
// adding Custom Options
container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
// custom Options Select Event
container.documentEditor.customContextMenuSelect = function (args) {
    // custom Options Functionality
    let id = container.documentEditor.element.id;
    switch (args.id) {
        case id + 'search_in_google':
            // To get the selected content as plain text
            let searchContent =
                container.documentEditor.selection.text;
            if (
                !container.documentEditor.selection.isEmpty &&
                /\S/.test(searchContent)
            ) {
                window.open('http://google.com/search?q=' +
searchContent);
            }
            break;
    }
};
// custom options hide/show functionality
container.documentEditor.customContextMenuBeforeOpen = function
(args) {
    let search = document.getElementById(args.ids[0]);
    search.style.display = 'none';
    let searchContent = container.documentEditor.selection.text;
    if (!container.documentEditor.selection.isEmpty &&
    /\S/.test(searchContent)) {
        search.style.display = 'block';
    }
};
}
</script>

```

CUSTOMIZE-CONTEXT-MENU.CS

Customize existing toolbar

How to customize existing toolbar in DocumentEditorContainer

DocumentEditorContainer allows to customize (add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can be defined by `CustomToolbarItemModel` and with existing items in `ToolbarItems` property. Newly added item click action can be defined in `Toolbarclick`.

- Show, Hide - Existing items can be shown or hidden using the [ToolBarItems](#) property. Pre-defined toolbar items are available with `ToolBarItem`.
- Enable, Disable - Toolbar items can be enabled or disabled using `enableItems`

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditorContainer("container").Created("onDocCreate").EnableToolbar(true).Render()
</div>
<script>
    function onDocCreate() {
        var container =
document.getElementById("container").ej2_instances[0];
        var toolItem = {
            prefixIcon: "e-de-ctnr-lock",
            tooltipText: "Disable Image",
            text: "Disable Image",
            id: "Custom"
        };
        container.toolbarItems = [toolItem, 'Undo', 'Redo', 'Separator',
'Image', 'Table', 'Hyperlink', 'Bookmark', 'Comments', 'TableOfContents',
'Separator', 'Header', 'Footer', 'PageSetup', 'PageNumber', 'Break',
'Separator', 'Find', 'Separator', 'LocalClipboard', 'RestrictEditing'];
        container.toolbarClick = function (args) {
            switch (args.item.id) {
                case 'Custom':
                    //Disable image toolbar item.
                    container.toolbar.enableItems(4, false);
                    break;
            }
        };
    }
</script>
```

CUSTOM-TOOLBAR.CS

Note: Default value of `ToolBarItems` is ['New', 'Open', 'Separator', 'Undo', 'Redo', 'Separator', 'Image', 'Table', 'Hyperlink', 'Bookmark', 'Comments', 'TableOfContents', 'Separator', 'Header', 'Footer', 'PageSetup', 'PageNumber', 'Break', 'Separator', 'Find', 'Separator', 'LocalClipboard', 'RestrictEditing'].

Change document view

How to change the document view in DocumentEditor component

DocumentEditor allows to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#)

CSHTML

```
@Html.EJS().DocumentEditor("container").LayoutType(Syncfusion.EJ2.DocumentEditor.LayoutType.Continuous).Render()
```

WEB-LAYOUT.CS

Note: Default value of [layoutType](#) in DocumentEditor component is [Pages](#).

How to change the document view in DocumentEditorContainer component

DocumentEditorContainer component allows to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#)

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").LayoutType(Syncfusion.EJ2.DocumentEditor.LayoutType.Continuous).Render()
```

WEB-LAYOUT.CS

Note: Default value of [layoutType](#) in DocumentEditorContainer component is [Pages](#).

How to open a default document in DocumentEditor when initialized

This article explains how to open a default document when DocumentEditor & DocumentEditorContainer is initialized.

Opening a default document in DocumentEditor

Using `open` method in Document editor allows to open the Document in sfdt format. To open the document by default, call the open method in the `created` event of Document editor which gets triggered once the control is created.

CSHTML

```
@Html.EJS().DocumentEditor("container").Created("onCreate").Render()  
<script>  
    var container;  
    function onCreate() {  
        var documenteditorElement = document.getElementById("container");  
        container = documenteditorElement.ej2_instances[0];  
        var sfdt = { "sections": [{ "sectionFormat": { "pageWidth": 612,  
"pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72,  
"bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages":  
false, "headerDistance": 36, "footerDistance": 36, "bidi": false },  
"blocks": [{ "paragraphFormat": { "afterSpacing": 30, "styleName": "Heading  
1", "listFormat": {} }, "characterFormat": {}, "inlines": [{  
"characterFormat": {}, "text": "Adventure Works Cycles" }] }],  
"headersFooters": { "header": { "blocks": [{ "paragraphFormat": {  
"listFormat": {} }, "characterFormat": {}, "inlines": [] } ] }, "footer": {  
"blocks": [{ "paragraphFormat": { "listFormat": {} }, "characterFormat": {},  
"inlines": [] } ] } }, "characterFormat": { "bold": false, "italic":  
false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None",
```

```

"strikethrough": "None", "baselineAlignment": "Normal", "highlightColor":
"NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi":
"Calibri", "allCaps": false }, "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 0, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "listFormat": {}, "bidi": false },
"defaultTabWidth": 36, "trackChanges": false, "enforcement": false,
"hashValue": "", "saltValue": "", "formatting": false, "protectionType":
"NoProtection", "dontUseHTMLParagraphAutoSpacing": false,
"formFieldShading": true, "styles": [{ "name": "Normal", "type":
"Paragraph", "paragraphFormat": { "lineSpacing": 1.149999976158142,
"lineSpacingType": "Multiple", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri", "next": "Normal" }, { "name": "Default Paragraph
Font", "type": "Character", "characterFormat": {} }, { "name": "Heading 1
Char", "type": "Character", "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 1", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 12, "afterSpacing": 0, "outlineLevel":
"Level1", "listFormat": {} }, "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 1 Char", "next": "Normal" }, { "name": "Heading 2
Char", "type": "Character", "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 2", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 2, "afterSpacing": 6, "outlineLevel":
"Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 2 Char", "next": "Normal" }, { "name": "Heading
3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {} },
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Normal", "link": "Heading 3 Char",
"next": "Normal" }, { "name": "Heading 3 Char", "type": "Character",
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {} },
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Normal", "link": "Heading 4 Char",
"next": "Normal" }, { "name": "Heading 4 Char", "type": "Character",
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {} },
"characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496"
}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, {
"name": "Heading 5 Char", "type": "Character", "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph",
"paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent":
0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0,

```

```

"lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple",
"outlineLevel": "Level6", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#1F3763" }, "basedOn":
"Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6
Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }],
"lists": [], "abstractLists": [], "comments": [], "revisions": [],
"customXml": [] };
    // open the default document.
    container.open(JSON.stringify(sfdt));
}
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

Opening a default document in DocumentEditorContainer

To open the document by default, call the open method in the **created** event of Document editor container which gets triggered once the control is created.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreate").EnableToolbar(true).Render()
<script>
    var container;
    function onCreate() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        var sfdt = { "sections": [{ "sectionFormat": { "pageWidth": 612,
"pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72,
"bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages":
false, "headerDistance": 36, "footerDistance": 36, "bidi": false },
"blocks": [{ "paragraphFormat": { "afterSpacing": 30, "styleName": "Heading
1", "listFormat": {} }, "characterFormat": {}, "inlines": [{
"characterFormat": {}, "text": "Adventure Works Cycles" }] }],
"headersFooters": { "header": { "blocks": [{ "paragraphFormat": {
"listFormat": {} }, "characterFormat": {}, "inlines": [] } ] }, "footer": {
"blocks": [{ "paragraphFormat": { "listFormat": {} }, "characterFormat": {},
"inlines": [] } ] } } ], "characterFormat": { "bold": false, "italic":
false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None",
"strikethrough": "None", "baselineAlignment": "Normal", "highlightColor":
"NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi":
"Calibri", "allCaps": false }, "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 0, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "listFormat": {}, "bidi": false },
"defaultTabWidth": 36, "trackChanges": false, "enforcement": false,
"hashValue": "", "saltValue": "", "formatting": false, "protectionType":
"NoProtection", "dontUseHTMLParagraphAutoSpacing": false,

```

```

"formFieldShading": true, "styles": [{ "name": "Normal", "type":
"Paragraph", "paragraphFormat": { "lineSpacing": 1.149999976158142,
"lineSpacingType": "Multiple", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri" }, "next": "Normal" }, { "name": "Default Paragraph
Font", "type": "Character", "characterFormat": {} }, { "name": "Heading 1
Char", "type": "Character", "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 1", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 12, "afterSpacing": 0, "outlineLevel":
"Level1", "listFormat": {} }, "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 1 Char", "next": "Normal" }, { "name": "Heading 2
Char", "type": "Character", "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 2", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 2, "afterSpacing": 6, "outlineLevel":
"Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 2 Char", "next": "Normal" }, { "name": "Heading
3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {} },
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Normal", "link": "Heading 3 Char",
"next": "Normal" }, { "name": "Heading 3 Char", "type": "Character",
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {} },
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Normal", "link": "Heading 4 Char",
"next": "Normal" }, { "name": "Heading 4 Char", "type": "Character",
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {} },
"characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496"
}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, {
"name": "Heading 5 Char", "type": "Character", "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph",
"paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent":
0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0,
"lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple",
"outlineLevel": "Level6", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#1F3763" }, "basedOn":
"Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6
Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }],
"lists": [], "abstractLists": [], "comments": [], "revisions": [],
"customXml": [] };
    // open the default document.

```

```

        container.documentEditor.open(JSON.stringify(sfdt));
    }
</script>

```

DOCUMENT-EDITOR.CS

```

public ActionResult Default()
{
    return View();
}

```

How to open a document in read only mode by default in Document Editor

This article explains how to open a document in read only mode by default in Document Editor & Document Editor Container.

Opening a document in read only mode by default in Document Editor

Using [isReadOnly](#) property in Document editor allows to enable or disable read only mode in the document editor.

CSHTML

```

<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnablePrint(true).EnableSelection(true).EnableSfdtExport(true).Created("onC
reated").Render()
</div>
<script>
    var documenteditor;
    function onCreate() {
        var documenteditorElement = document.getElementById("container");
        documenteditor = documenteditorElement.ej2_instances[0];
        // load your default document here
        var data = { "sections": [{ "sectionFormat": { "pageWidth": 612,
"pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72,
"bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages":
false, "headerDistance": 36, "footerDistance": 36, "bidi": false },
"blocks": [{ "paragraphFormat": { "afterSpacing": 30, "styleName": "Heading
1", "listFormat": {} }, "characterFormat": {}, "inlines": [{
"characterFormat": {}, "text": "Adventure Works Cycles" }] }],
"headersFooters": { "header": { "blocks": [{ "paragraphFormat": {
"listFormat": {} }, "characterFormat": {}, "inlines": [] } ] }, "footer": {
"blocks": [{ "paragraphFormat": { "listFormat": {} }, "characterFormat": {},
"inlines": [] } ] } }, "characterFormat": { "bold": false, "italic":
false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None",
"strikethrough": "None", "baselineAlignment": "Normal", "highlightColor":
"NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi":
"Calibri", "allCaps": false }, "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 0, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "listFormat": {}, "bidi": false },
"defaultTabWidth": 36, "trackChanges": false, "enforcement": false,
"hashValue": "", "saltValue": "", "formatting": false, "protectionType":
"NoProtection", "dontUseHTMLParagraphAutoSpacing": false,

```



```

"formFieldShading": true, "styles": [{ "name": "Normal", "type":
"Paragraph", "paragraphFormat": { "lineSpacing": 1.149999976158142,
"lineSpacingType": "Multiple", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri" }, "next": "Normal" }, { "name": "Default Paragraph
Font", "type": "Character", "characterFormat": {} }, { "name": "Heading 1
Char", "type": "Character", "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 1", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 12, "afterSpacing": 0, "outlineLevel":
"Level1", "listFormat": {} }, "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 1 Char", "next": "Normal" }, { "name": "Heading 2
Char", "type": "Character", "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 2", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 2, "afterSpacing": 6, "outlineLevel":
"Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 2 Char", "next": "Normal" }, { "name": "Heading
3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {} },
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Normal", "link": "Heading 3 Char",
"next": "Normal" }, { "name": "Heading 3 Char", "type": "Character",
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {} },
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Normal", "link": "Heading 4 Char",
"next": "Normal" }, { "name": "Heading 4 Char", "type": "Character",
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {} },
"characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496"
}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, {
"name": "Heading 5 Char", "type": "Character", "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph",
"paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent":
0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0,
"lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple",
"outlineLevel": "Level6", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#1F3763" }, "basedOn":
"Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6
Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }],
"lists": [], "abstractLists": [], "comments": [], "revisions": [],
"customXml": [] };
// Open the default document

```

```
documenteditor.open(JSON.stringify(data));
//Enable read only mode.
documenteditor.isReadOnly = true;
}
</script>
```

READ-ONLY.CS

Opening a document in ready only mode by default in Document Editor Container

Using [restrictEditing](#) property in Document editor container allows to enable or disable read only mode in the document editor.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // load your default document here
        var data = { "sections": [{ "sectionFormat": { "pageWidth": 612,
"pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72,
"bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages":
false, "headerDistance": 36, "footerDistance": 36, "bidi": false },
"blocks": [{ "paragraphFormat": { "afterSpacing": 30, "styleName": "Heading
1", "listFormat": {} }, "characterFormat": {}, "inlines": [{
"characterFormat": {}, "text": "Adventure Works Cycles" }] }],
"headersFooters": { "header": { "blocks": [{ "paragraphFormat": {
"listFormat": {} }, "characterFormat": {}, "inlines": [] } ] }, "footer": {
"blocks": [{ "paragraphFormat": { "listFormat": {} }, "characterFormat": {},
"inlines": [] } ] } }, "characterFormat": { "bold": false, "italic":
false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None",
"strikethrough": "None", "baselineAlignment": "Normal", "highlightColor":
"NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi":
"Calibri", "allCaps": false }, "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 0, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "listFormat": {}, "bidi": false },
"defaultTabWidth": 36, "trackChanges": false, "enforcement": false,
"hashValue": "", "saltValue": "", "formatting": false, "protectionType":
"NoProtection", "dontUseHTMLParagraphAutoSpacing": false,
"formFieldShading": true, "styles": [{ "name": "Normal", "type":
"Paragraph", "paragraphFormat": { "lineSpacing": 1.149999976158142,
"lineSpacingType": "Multiple", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri" }, "next": "Normal" }, { "name": "Default Paragraph
Font", "type": "Character", "characterFormat": {} }, { "name": "Heading 1
Char", "type": "Character", "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 1", "type": "Paragraph",
```

```

"paragraphFormat": { "beforeSpacing": 12, "afterSpacing": 0, "outlineLevel":
"Level1", "listFormat": {} }, "characterFormat": { "fontSize": 16,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 1 Char", "next": "Normal" }, { "name": "Heading 2
Char", "type": "Character", "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 2", "type": "Paragraph",
"paragraphFormat": { "beforeSpacing": 2, "afterSpacing": 6, "outlineLevel":
"Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13,
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn":
"Normal", "link": "Heading 2 Char", "next": "Normal" }, { "name": "Heading
3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {} },
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Normal", "link": "Heading 3 Char",
"next": "Normal" }, { "name": "Heading 3 Char", "type": "Character",
"characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light",
"fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {} },
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Normal", "link": "Heading 4 Char",
"next": "Normal" }, { "name": "Heading 4 Char", "type": "Character",
"characterFormat": { "italic": true, "fontFamily": "Calibri Light",
"fontColor": "#2F5496" }, "basedOn": "Default Paragraph Font" }, { "name":
"Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0,
"rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left",
"beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947,
"lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {} },
"characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496"
}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, {
"name": "Heading 5 Char", "type": "Character", "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default
Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph",
"paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent":
0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0,
"lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple",
"outlineLevel": "Level6", "listFormat": {} }, "characterFormat": {
"fontFamily": "Calibri Light", "fontColor": "#1F3763" }, "basedOn":
"Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6
Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#1F3763" }, "basedOn": "Default Paragraph Font" }],
"lists": [], "abstractLists": [], "comments": [], "revisions": [],
"customXml": [] };

    // Open the default document
    documenteditor.open(JSON.stringify(data));
    //Enable read only mode.
    container.restrictEditing = true;
}
</script>

```

READ-ONLY.CS

Note: You can use the [restrictEditing](#) in [DocumentEditorContainerComponent](#) and [isReadOnly](#) in [DocumentEditorComponent](#) based on your requirement to change component to read only mode.

Open a document from URL

How to open a document from URL in DocumentEditor

This article explains how to open a document from URL in DocumentEditor.

CSHTML

```
<div>
    @Html.EJS().Button("import").Content("Open
Document").Click("onClick").Render()

    @Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
</div>
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
    }
    function onClick() {
        let http = new XMLHttpRequest();
        //add your url in which you want to open document inside the ""
        let content = { fileUrl: "" };
        let baseUrl = "/api/documenteditor/ImportFileURL";
        http.open("POST", baseUrl, true);
        http.setRequestHeader("Content-Type", "application/json;charset=UTF-
8");
        http.onreadystatechange = () => {
            if (http.readyState === 4) {
                if (http.status === 200 || http.status === 304) {
                    //open the SFDT text in Document Editor
                    container.documentEditor.open(http.responseText);
                }
            }
        };
        http.send(JSON.stringify(content));
    }
</script>
```

OPEN-BY-URL.CS

```
`typescript
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
import {
  DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
export class App extends React.Component<{}, {}> {
  container: DocumentEditorContainerComponent;
  public contentChanged:boolean=false;
  onClick():void {
    let http: XMLHttpRequest = new XMLHttpRequest();
    //add your url in which you want to open document inside the ""
    let content = { fileUrl: "" };
    let baseUrl: string = "/api/documenteditor/ImportFileURL";
    http.open("POST", baseUrl, true);
    http.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    http.onreadystatechange = () => {
      if (http.readyState === 4) {
        if (http.status === 200 || http.status === 304) {
          //open the SFDT text in Document Editor
          container.documentEditor.open(http.responseText);
        }
      }
    };
    http.send(JSON.stringify(content));
  }
  render() {
    return (
      <div>
        <button id='import' onClick={this.onClick.bind(this)}>Import</button>
        <DocumentEditorContainerComponent id="container" ref={{scope} => { this.container = scope; }}
          height={'590px'}
          serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
          enableToolbar={true}
        />
      </div>
    );
  }
}
```

```
</div>
);
}
}
ReactDOM.render(<App />, document.getElementById('root'));
、

`csharp
[AcceptVerbs("Post")]
public string ImportFileURL([FromBody]FileUrlInfo param)
{
    try {
        using(WebClient client = new WebClient())
        {
            MemoryStream stream = new MemoryStream(client.DownloadData(param.fileUrl));
            WordDocument document = WordDocument.Load(stream, FormatType.Docx);
            string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
            document.Dispose();
            stream.Dispose();
            return json;
        }
    }
    catch (Exception) {
        return "";
    }
}

public class FileUrlInfo {
    public string fileUrl { get; set; }
    public string Content { get; set; }
}
、
```

Deploy Document Editor component for Mobile

Document editor component for Mobile

At present, Document editor component is not responsive for mobile, and the editing functionalities aren't ensured in mobile browsers. Whereas it works properly as a document viewer in mobile browsers.

Hence, it is recommended to switch the Document editor component as read-only in mobile browsers.

Also, invoke `fitPage` method with `FitPageWidth` parameter in document change event, such as to display one full page by adjusting the zoom factor.

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnablePrint(true).EnableSelection(true).EnableSfdtExport(true).DocumentChange(
"onDocumentChange").Render()
</div>
<script>
    function onDocumentChange() {
        var container =
document.getElementById("container").ej2_instances[0];
        //To detect the device
        var isMobileDevice = /Android|Windows
Phone|webOS/i.test(navigator.userAgent);
        if (isMobileDevice) {
            container.restrictEditing = true;
            setTimeout(() => {
                container.documentEditor.fitPage("FitPageWidth");
            }, 50);
        }
        else {
            container.restrictEditing = false;
        }
    }
</script>
```

MOBILE-VIEW.CS

Note: You can use the [restrictEditing](#) in DocumentEditorContainer and [isReadOnly](#) in DocumentEditor based on your requirement to change component to read only mode.

How to disable optimized text measuring in Document Editor component

Starting from v19.3.0.x, the accuracy of text size measurements in Document editor is improved such as to match Microsoft Word pagination for most Word documents. This improvement is included as default behavior along with an optional API `enableOptimizedTextMeasuring` in Document editor settings.

If you want the Document editor component to retain the document pagination (display page-by-page) behavior like v19.2.0.x and older versions. Then, you can disable this optimized text measuring improvement, by setting `false` to `enableOptimizedTextMeasuring` property of Document Editor component.

Disable optimized text measuring in `DocumentEditorContainer` instance

The following example code illustrates how to disable optimized text measuring improvement in `DocumentEditorContainer` instance.

CSHTML

```
<div id="documenteditor" style="width:100%;height:100%">

@Html.EJS().DocumentEditor("container").IsReadOnly(false).EnableEditor(true)
.EnablePrint(true).EnableSelection(true).EnableSfdtExport(true).DocumentEdit
orSettings("settings").Render()
</div>
<script>
    var settings = { enableOptimizedTextMeasuring: false };
</script>
```

OPTIMIZED-TEXT.CS*Disable optimized text measuring in `DocumentEditor` instance*

The following example code illustrates how to disable optimized text measuring improvement in `DocumentEditor` instance.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").EnableToolbar(true).Documen
tEditorSettings("settings").Render()
<script>
    var settings = { enableOptimizedTextMeasuring: false };
</script>
```

OPTIMIZED-TEXT.CS*How to get the selected content in Document Editor component*

You can get the selected content from the React Document Editor component as plain text and SFDT (rich text).

Get the selected content as plain text

You can use `text` API to get the selected content as plain text from React Document Editor component.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
    }
</script>
```



```

documenteditor = container.documentEditor;
// creating Custom Options
let menuItems = [
    {
        text: 'Search In Google',
        id: 'search_in_google',
        iconCss: 'e-icons e-de-ctnr-find',
    },
];
// adding Custom Options
container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
// custom Options Select Event
container.documentEditor.customContextMenuSelect =function (args){
    // custom Options Functionality
    let id = container.documentEditor.element.id;
    switch (args.id) {
        case id + 'search_in_google':
            // To get the selected content as plain text
            let searchContent =
                container.documentEditor.selection.text;
            if (!container.documentEditor.selection.isEmpty &&
                /\S/.test(searchContent)) {
                window.open('http://google.com/search?q=' + searchContent);
            }
            break;
        }
    };
}
</script>

```

GET-TEXT.CS

You can add the following custom options using this API,

- Save or export the selected text as text file.
- Search the selected text in Google or other search engines.
- Show synonyms for the selected word in context menu and replace with selected synonym using the setter method of same API.

Get the selected content as SFDT (rich text)

You can use `sfdt` API to get the selected content as plain text from React Document Editor component.

CSHTML

```

<div>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
</div>
<script>
    var documenteditor;

```

```

var container;
function onCreate() {
    var documenteditorElement = document.getElementById("container");
    container = documenteditorElement.ej2_instances[0];
    documenteditor = container.documentEditor;
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To select all the content in document
    container.documentEditor.selection.selectAll();
    // Insert bookmark to selected content
    container.documentEditor.editor.insertBookmark('Bookmark1');
    //Select the bookmark
    container.documentEditor.selection.selectBookmark('Bookmark1');
    // To get the selected content as sfdt
    let selectedContent = container.documentEditor.selection.sfdt;
    // Insert the sfdt content in cursor position using paste API
    container.documentEditor.editor.paste(selectedContent);
}
</script>

```

GET-SFDT.CS

You can add the following custom options using this API,

- Save or export the selected content as SFDT file.
- Get the content of a bookmark in Word document as SFDT by selecting a bookmark using `selectbookmark` API.
- Create template content that can be inserted to multiple documents in cursor position using `paste` API.

How to set the default character, paragraph and section format in Document Editor component
 You can set the default character format, paragraph format and section format in Document editor.

Set the default character format

You can use `setDefaultCharacterFormat` method to set the default character format. For example, Document editor default font size is 11 and you can change it as any valid value.

CSHTML

```

<div>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
</div>
<script>
    var documenteditor;
    var container;
    function onCreate() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        container.documentEditor.setDefaultCharacterFormat({fontSize: 20});
    }
</script>

```

```
}  
</script>
```

CHARACTER-FORMAT-FONT.CS

Similarly, you can change the required `CharacterFormatProperties` default value.

CSHTML

```
<div>  
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable  
Toolbar(true).Render()  
</div>  
<script>  
    var documenteditor;  
    var container;  
    function onCreated() {  
        var documenteditorElement = document.getElementById("container");  
        container = documenteditorElement.ej2_instances[0];  
        documenteditor = container.documentEditor;  
        var defaultCharacterFormat = {  
            bold: false,  
            italic: false,  
            baselineAlignment: 'Normal',  
            underline: 'None',  
            fontColor: "#000000",  
            fontFamily: 'Algerian',  
            fontSize: 12  
        };  
  
        container.documentEditor.setDefaultCharacterFormat(defaultCharacterFormat);  
    }  
</script>
```

CHARACTER-FORMAT.CS

Set the default paragraph format

You can use `setDefaultParagraphFormat` API to set the default paragraph format. You can change the required `ParagraphFormatProperties` default value.

CSHTML

```
<div>  
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable  
Toolbar(true).Render()  
</div>  
<script>  
    var documenteditor;  
    var container;
```

```
function onCreated() {
    var documenteditorElement = document.getElementById("container");
    container = documenteditorElement.ej2_instances[0];
    documenteditor = container.documentEditor;
    var defaultParagraphFormat = {
        beforeSpacing: 8,
        lineSpacing: 1.5,
        leftIndent: 24,
        textAlignment: "Center"
    };

    container.documentEditor.setDefaultParagraphFormat(defaultParagraphFormat);
}
</script>
```

PARAGRAPH-FORMAT.CS

Set the default section format

You can use `setDefaultSectionFormat` API to set the default section format. You can change the required `SectionFormatProperties` default value.

CSHTML

```
<div>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
</div>
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        var defaultSectionFormat = {
            pageWidth: 500,
            pageHeight: 800,
            headerDistance: 56,
            footerDistance: 48,
            leftMargin: 12,
            rightMargin: 12,
            topMargin: 0,
            bottomMargin: 0
        };

        container.documentEditor.setDefaultSectionFormat(defaultSectionFormat);
    }
</script>
```

SECTION-FORMAT.CS

How to show and hide spinner while opening document in React Document Editor component
Using [spinner](#) component, you can show or hide spinner while opening document in DocumentEditor.

```
`typescript
// showSpinner() will make the spinner visible
showSpinner(document.getElementById('container'));

// hideSpinner() method used hide spinner
hideSpinner(document.getElementById('container'));
`
```

CSHTML

```
<div>
    @Html.EJS().Button("import").Content("Load
    Document").Click("onClick").Render()

    @Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
    Toolbar(true).Render()
</div>
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        ej.popups.createSpinner({
            // Specify the target for the spinner to show
            target: document.getElementById('container')
        });
    }
    function onClick() {
        // load your default document here
        let data =
        '{ "sections": [{ "sectionFormat": { "pageWidth": 612, "pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72, "bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages": false, "headerDistance": 36, "footerDistance": 36, "bidi": false }, "blocks": [{ "paragraphFormat": { "afterSpacing": 30, "styleName": "Heading 1", "listFormat": {}, "characterFormat": {}, "inlines": [{ "characterFormat": {}, "text": "Adventure Works Cycles" } ] } }, { "headersFooters": { "header": { "blocks": [{ "paragraphFormat": { "listFormat": {}, "characterFormat": {}, "inlines": [ ] } } ], "footer": { "blocks": [{ "paragraphFormat": { "listFormat": {}, "characterFormat": {}, "inlines": [ ] } } } } }, { "characterFormat": { "bold": false, "italic": false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignment": "Normal", "highlightColor": "NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi": "Calibri", "allCaps": false }, "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "listForm
```

```

at": {}, "bidi": false, "defaultTabWidth": 36, "trackChanges": false, "enforcement":
: false, "hashValue": "", "saltValue": "", "formatting": false, "protectionType": "No
Protection", "dontUseHTMLParagraphAutoSpacing": false, "formFieldShading": true,
"styles": [ { "name": "Normal", "type": "Paragraph", "paragraphFormat": { "lineSpacin
g": 1.149999976158142, "lineSpacingType": "Multiple", "listFormat": {} }, "characte
rFormat": { "fontFamily": "Calibri", "next": "Normal" }, { "name": "Default
Paragraph Font", "type": "Character", "characterFormat": {} }, { "name": "Heading 1
Char", "type": "Character", "characterFormat": { "fontSize": 16, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph
Font" }, { "name": "Heading
1", "type": "Paragraph", "paragraphFormat": { "beforeSpacing": 12, "afterSpacing": 0
, "outlineLevel": "Level1", "listFormat": {} }, "characterFormat": { "fontSize": 16, "
fontFamily": "Calibri
Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 1
Char", "next": "Normal" }, { "name": "Heading 2
Char", "type": "Character", "characterFormat": { "fontSize": 13, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph
Font" }, { "name": "Heading
2", "type": "Paragraph", "paragraphFormat": { "beforeSpacing": 2, "afterSpacing": 6,
"outlineLevel": "Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13, "f
ontFamily": "Calibri
Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 2
Char", "next": "Normal" }, { "name": "Heading
3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level3", "listFormat": {} }, "characterFormat": { "fontSize": 12, "fontFamily": "Cali
bri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 3
Char", "next": "Normal" }, { "name": "Heading 3
Char", "type": "Character", "characterFormat": { "fontSize": 12, "fontFamily": "Cali
bri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph
Font" }, { "name": "Heading
4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level4", "listFormat": {} }, "characterFormat": { "italic": true, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 4
Char", "next": "Normal" }, { "name": "Heading 4
Char", "type": "Character", "characterFormat": { "italic": true, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph
Font" }, { "name": "Heading
5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level5", "listFormat": {} }, "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 5
Char", "next": "Normal" }, { "name": "Heading 5
Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph
Font" }, { "name": "Heading
6", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level6", "listFormat": {} }, "characterFormat": { "fontFamily": "Calibri
Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal" }, { "name": "Heading 6
Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri

```

```
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXm
l": []}';

    // showSpinner() will make the spinner visible
    ej.popups.showSpinner(document.getElementById('container'));
    // Open the default document
    container.documentEditor.open(data);
    setInterval(function () {
        // hideSpinner() method used hide spinner
        ej.popups.hideSpinner(document.getElementById('container'));
    }, 5000);
}
</script>
```

SPINNER.CS

Note: In above example, we have used setInterval to hide spinner, just for demo purpose.

How to change height and width of Document Editor component

This article explains how to change height and width of Document editor.

Change height of Document Editor

DocumentEditorContainer initially renders with default height. You can change the height of document editor using **height** property, the value which is in pixel.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Height('590px').EnableToolb
ar(true).Render()
```

CHANGE-HEIGHT.CS

Similarly, you can use **height** property for DocumentEditor also.

Change width of Document Editor

DocumentEditorContainer initially renders with default width. You can change the width of document editor using **width** property, the value which is in percent.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Width('100%').EnableToolb
ar(true).Render()
```

CHANGE-WIDTH.CS

Similarly, you can use `width` property for DocumentEditor also.

[Resize Document Editor](#)

Using `resize` method, you change height and width of Document editor.

CSHTML

```
<div>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
</div>
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        setInterval(() => {
            updateDocumentEditorSize();
        }, 100);
        //Adds event listener for browser window resize event.
        window.addEventListener('resize', onWindowResize);
    }
    function onWindowResize() {
        //Resizes the document editor component to fit full browser window
        automatically whenever the browser resized.
        updateDocumentEditorSize();
    }
    function updateDocumentEditorSize() {
        //Resizes the document editor component to fit full browser window.
        var windowWidth = window.innerWidth;
        var windowHeight = window.innerHeight;
        container.resize(windowWidth, windowHeight);
    }
</script>
```

RESIZE.CS

[How to export the document as PDF in React Document Editor](#)

This article explains how to export the document as PDF format. You can export the document as PDF in following ways:

[Export the document as PDF in client-side](#)

Use [pdf export component](#) in application level to export the document as PDF using `exportasimage` API. Here, all pages will be converted to image and inserted as PDF pages (works like print as PDF). There is one limitation, the text can't be searched because the PDF is exported as image.

Note: You can install the PDF export packages from this [link](#).

CSHTML


```

@Html.EJS().Button("element").Content("Using Pdf
Export").Click("exportClientSide").Render()
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
    }
    //Pdf document exported using pdf export
    function exportClientSide() {
        let pdfdocument = new ej.pdfexport.PdfDocument();
        let count = container.documentEditor.pageCount;

        container.documentEditor.documentEditorSettings.printDevicePixelRatio = 2;
        let loadedPage = 0;
        for (let i = 1; i <= count; i++) {
            setTimeout(() => {
                let format = 'image/jpeg';
                // Getting pages as image
                let image = container.documentEditor.exportAsImage(i,
format);
                image.onload = function () {
                    let imageHeight = parseInt(
                        image.style.height.toString().replace('px', ''))
                    );
                    let imageWidth = parseInt(
                        image.style.width.toString().replace('px', ''))
                    );
                    let section = pdfdocument.sections.add();
                    let settings = new ej.pdfexport.PdfPageSettings(0);
                    if (imageWidth > imageHeight) {
                        settings.orientation = PdfPageOrientation.Landscape;
                    }
                    settings.size = new ej.pdfexport.SizeF(imageWidth,
imageHeight);

                    (section).setPageSettings(settings);
                    let page = section.pages.add();
                    let graphics = page.graphics;
                    let imageStr =
image.src.replace('data:image/jpeg;base64,', '');
                    let pdfImage = new ej.pdfexport.PdfBitmap(imageStr);
                    graphics.drawImage(pdfImage, 0, 0, imageWidth,
imageHeight);

                    loadedPage++;
                    if (loadedPage == count) {
                        // Exporting the document as pdf
                        pdfdocument.save(
                            (container.documentEditor.documentName === ''
                                ? 'sample'
                                : container.documentEditor.documentName) +
                            '.pdf'
                        );
                    }
                }
            });
        }
    }

```

```

        };
    }, 500);
}
}
</script>

```

EXPORT-PDF-CLIENT.CS

Export document as PDF in server-side using Syncfusion DocIO

With the help of [Syncfusion DocIO](#), you can export the document as PDF in server-side. Here, you can search the text.

The following way illustrates how to convert the document as PDF:

- Using `serialize` API, convert the document as Sfdt and send it to server-side.

CSHTML

```

@Html.EJS().Button("element").Content("Export").Click("exportServerSide").Render()
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
    }
    function exportServerSide() {
        let http = new XMLHttpRequest();
        // Replace your running web service Url here
        http.open('POST', '/api/documenteditor/ExportPdf');
        http.setRequestHeader('Content-Type', 'application/json;charset=UTF-
8');
        http.responseType = 'json';
        //Serialize document content as SFDT.
        let sfdt = { content: container.documentEditor.serialize() };
        //Send the sfdt content to server side.
        http.send(JSON.stringify(sfdt));
    }
</script>

```

EXPORT-PDF-SERVER.CS

- Using `Save` API in server-side, you can convert the sfdt to stream.

- Finally, convert the stream to PDF using `Syncfusion.DocIO.Renderer.Net.Core` library.

```
`csharp
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("ExportPdf")]
public void ExportPdf([FromBody]SaveParameter data)
{
    // Converts the sfdt to stream
    Stream document = WordDocument.Save(data.content, FormatType.Docx);
    Syncfusion.DocIO.DLS.WordDocument doc = new Syncfusion.DocIO.DLS.WordDocument(document,
    Syncfusion.DocIO.FormatType.Docx);
    //Instantiation of DocIO.Renderer for Word to PDF conversion
    DocIO.Renderer render = new DocIO.Renderer();
    //Converts Word document into PDF document
    PdfDocument pdfDocument = render.ConvertToPDF(doc);
    // Saves the document to server machine file system, you can customize here to save into databases or
    file servers based on requirement.
    FileStream fileStream = new FileStream("sample.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
    //Saves the PDF file
    pdfDocument.Save(fileStream);
    pdfDocument.Close();
    fileStream.Close();
    document.Close();
}
```

Get the complete working sample in this [link](#).

How to customize the font family drop down in React Document Editor component

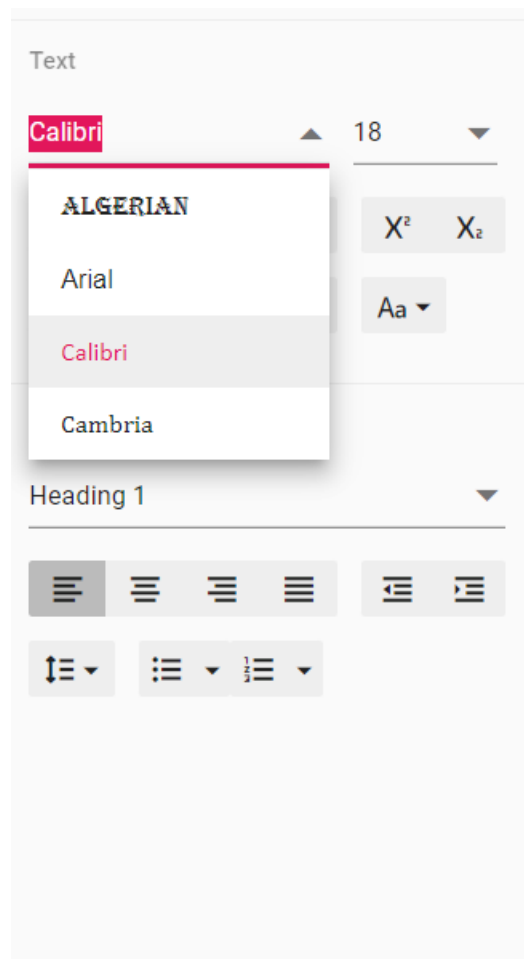
Document editor provides options to customize the font family drop down list values using `fontFamilies` in Document editor settings. Fonts which are added in `fontFamilies` of `documentEditorSettings` will be displayed on font drop down list of text properties pane and font dialog.

Similarly, you can use `documentEditorSettings` property for DocumentEditor also.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").EnableToolbar(true).DocumentEditorSettings("settings").Render()
<script>
    var settings = { fontFamilies: ['Algerian', 'Arial', 'Calibri', 'Cambria'] };
</script>
```

FONT-FAMILY.CS



How to auto save the document of Document Editor component into AWS S3

This article explains how to autosave the document in AWS S3. You can automatically save the edited content in regular intervals of time. It helps to reduce the risk of data loss by saving an open document automatically at customized intervals.

- In the client-side, using content change event, the edited content can be automatically saved in regular intervals of time. Based on `contentChanged` boolean, the document send as Docx format to server-side using `saveAsBlob` method.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreate").EnableToolbar(true).Render()
<script>
    var container;
    var containerPanel;
    var contentChanged = false;
    document.addEventListener('DOMContentLoaded', function () {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        container.contentChange=function(){
            contentChanged = true;
        }
    });
    function onCreate() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        setInterval(() => {
            if (contentChanged) {
                //You can save the document as below
                container.documentEditor.saveAsBlob('Docx').then((blob) => {
                    console.log('Saved sucessfully');
                    let exportedDocument = blob;
                    //Now, save the document where ever you want.
                    let formData = new FormData();
                    formData.append('fileName', 'sample.docx');
                    formData.append('data', exportedDocument);
                    /* tslint:disable */
                    var req = new XMLHttpRequest();
                    // Replace your running Url here
                    req.open(
                        'POST',
                        'http://localhost:62869/api/documenteditor/SaveToS3',
                        true
                    );
                    req.onreadystatechange = () => {
                        if (req.readyState === 4) {
                            if (req.status === 200 || req.status === 304) {
                                console.log('Saved sucessfully');
                            }
                        }
                    };
                    req.send(formData);
                });
                contentChanged = false;
            }
        }, 1000);
    }
</script>

```

AUTO-SAVE.CS

- Configure the access key and secret key in `web.config` file and register profile in `startup.cs`.

In `web.config`, add key like below format:

```
`c#  
<appSettings>  
<add key="AWSProfileName" value="sync_development" />  
<add key="AWSAccessKey" value="" />  
<add key="AWSSecretKey" value="" />  
</appSettings>  
,
```

In `startup.cs`, register profile in below format:

```
`c#  
Amazon.Util.ProfileManager.RegisterProfile("sync_development", "", "");  
,
```

- In server-side, Receives the stream content from client-side and process it to save the document in aws s3. Add Web API in controller file like below to save the document in aws s3.

```
`c#  
[AcceptVerbs("Post")]  
[HttpPost]  
[EnableCors("AllowAllOrigins")]  
[Route("SaveToS3")]  
public string SaveToS3()  
{  
    IFormFile file = HttpContext.Request.Form.Files[0];  
    Stream stream = new MemoryStream();  
    file.CopyTo(stream);  
    UploadFileStreamToS3(stream, "documenteditor", "", "GettingStarted.docx");  
    stream.Close();  
    return "Sucess";  
}  
  
public bool UploadFileStreamToS3(System.IO.Stream localFilePath, string bucketName, string  
subDirectoryInBucket, string fileNameInS3)
```

```

{
    AWSCredentials credentials = new StoredProfileAWSCredentials("sync_development");
    IAmazonS3 client = new AmazonS3Client(credentials, Amazon.RegionEndpoint.USEast1);
    TransferUtility utility = new TransferUtility(client);
    TransferUtilityUploadRequest request = new TransferUtilityUploadRequest();
    if (subDirectoryInBucket == "" || subDirectoryInBucket == null)
    {
        request.BucketName = bucketName; //no subdirectory just bucket name
    }
    else
    {
        // subdirectory and bucket name
        request.BucketName = bucketName + @"/" + subDirectoryInBucket;
    }
    request.Key = fileNameInS3; //file name up in S3
    request.InputStream = localFilePath;
    utility.Upload(request); //commencing the transfer
    return true; //indicate that the file was sent
}

```

Get the complete working sample in this [link](#).

How to retrieve the whole document and bookmark content as text in Document Editor component

You can get the bookmark or whole document content from the Document Editor component as plain text and SFDT (rich text).

Get the bookmark content as plain text

You can [selectBookmark] API to navigate to the bookmark and use [text] API to get the bookmark content as plain text from Document Editor component.

The following example code illustrates how to get the bookmark content as plain text.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
    }

```

```

// creating Custom Options
let menuItems = [
    {
        text: 'Search In Google',
        id: 'search_in_google',
        iconCss: 'e-icons e-de-ctnr-find',
    },
];
// adding Custom Options
container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
// custom Options Select Event
container.documentEditor.customContextMenuSelect =function (args){
    // custom Options Functionality
    let id = container.documentEditor.element.id;
    switch (args.id) {
        case id + 'search_in_google':
            // To get the selected content as plain text
            let searchContent =
                container.documentEditor.selection.text;
            if (!container.documentEditor.selection.isEmpty &&
/\S/.test(searchContent)) {
                window.open('http://google.com/search?q=' + searchContent);
            }
            break;
        }
    };
}
</script>

```

GET-TEXT.CS

To get the bookmark content as SFDT (rich text), check this [link](#)

Get the whole document content as text

You can use [text] API to get the whole document content as plain text from Document Editor component.

The following example code illustrates how to get the whole document content as plain text.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // creating Custom Options
        let menuItems = [

```



```

        {
            text: 'Search In Google',
            id: 'search_in_google',
            iconCss: 'e-icons e-de-ctnr-find',
        },
    ];
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
    // custom Options Select Event
    container.documentEditor.customContextMenuSelect =function (args){
        // custom Options Functionality
        let id = container.documentEditor.element.id;
        switch (args.id) {
            case id + 'search_in_google':
                // To get the selected content as plain text
                let searchContent =
                    container.documentEditor.selection.text;
                if (!container.documentEditor.selection.isEmpty &&
                    /\S/.test(searchContent)) {
                    window.open('http://google.com/search?q=' + searchContent);
                }
                break;
            }
        };
    }
}
</script>

```

GET-TEXT.CS

Get the whole document content as SFDT(rich text)

You can use [serialize] API to get the whole document content as SFDT string from Document Editor component.

The following example code illustrates how to get the whole document content as SFDT.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // creating Custom Options
        let menuItems = [
            {
                text: 'Search In Google',
                id: 'search_in_google',
                iconCss: 'e-icons e-de-ctnr-find',
            }
        ];
    }
}

```

```

    },
    ];
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
    // custom Options Select Event
    container.documentEditor.customContextMenuSelect =function (args){
    // custom Options Functionality
    let id = container.documentEditor.element.id;
    switch (args.id) {
        case id + 'search_in_google':
            // To get the selected content as plain text
            let searchContent =
                container.documentEditor.selection.text;
            if (!container.documentEditor.selection.isEmpty &&
/\S/.test(searchContent)) {
                window.open('http://google.com/search?q=' + searchContent);
            }
            break;
        }
    };
}
</script>

```

GET-TEXT.CS

Get the header content as text

You can use `[goToHeader]` API to navigate the selection to the header and then use `[text]` API to get the content as plain text.

The following example code illustrates how to get the header content as plain text.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        documenteditor = container.documentEditor;
        // creating Custom Options
        let menuItems = [
            {
                text: 'Search In Google',
                id: 'search_in_google',
                iconCss: 'e-icons e-de-ctnr-find',
            },
        ],
    };
    // adding Custom Options

```

```

        container.documentEditor.contextMenu.addCustomMenu(menuItems,
false);
        // custom Options Select Event
        container.documentEditor.customContextMenuSelect =function (args){
            // custom Options Functionality
            let id = container.documentEditor.element.id;
            switch (args.id) {
                case id + 'search_in_google':
                    // To get the selected content as plain text
                    let searchContent =
                        container.documentEditor.selection.text;
                    if (!container.documentEditor.selection.isEmpty &&
                        /\S/.test(searchContent)) {
                        window.open('http://google.com/search?q=' + searchContent);
                    }
                    break;
            }
        };
    }
</script>

```

GET-TEXT.CS

Similarly, you can use `[goToFooter]` API to navigate the selection to the footer and then use `[text]` API to get the content as plain text.

How to select and retrieve the word and paragraph in current cursor position in Document Editor component

You can get the current word or paragraph content from the Document Editor component as plain text and SFD (rich text).

Select and get the word in current cursor position

You can use `[selectCurrentWord]` API in selection module to select the current word at cursor position and use `[text]` API to get the selected content as plain text from Document Editor component.

The following example code illustrates how to select and get the current word as plain text.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        // To insert text in cursor position
        container.documentEditor.editor.insertText('Document editor');
        // To select the current word in document
        container.documentEditor.selection.selectCurrentWord();
        // To get the selected content as text
    }

```

```
var selectedContent = container.documentEditor.selection.text;
}
</script>
```

GET-WORD.CS

To get the bookmark content as SFDT (rich text), check this [link](#)

Select and get the paragraph in current cursor position

You can use [selectParagraph] API in selection module to select the current paragraph at cursor position and use [text] API or [sfdt] API to get the selected content as plain text or SFDT from Document Editor component.

The following example code illustrates how to select and get the current paragraph as SFDT.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
var documenteditor;
var container;
function onCreated() {
var documenteditorElement = document.getElementById("container");
container = documenteditorElement.ej2_instances[0];
// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');
// To select the current paragraph in document
container.documentEditor.selection.selectParagraph();
// To get the selected content as sfdt
var selectedContent = container.documentEditor.selection.sfdt;
}
</script>
```

GET-PARAGRAPH.CS

How to insert page number and navigate to specific page in Document Editor component

You can insert page number and navigate to specific page in Document Editor component by following ways.

Insert page number

You can use [insertPageNumber] API in editor module to insert the page number in current cursor position. By default, Page number will insert in Arabic number style. You can change it, by providing the number style in parameter.

Note: Currently, Documenteditor have options to insert page number at current cursor position.

The following example code illustrates how to insert page number in header.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        // To insert text in cursor position
        container.documentEditor.editor.insertText('Document editor');
        // To move the selection to header
        container.documentEditor.selection.goToHeader();
        // Insert page number in the current cursor position
        container.documentEditor.editor.insertPageNumber();
    }
</script>
```

INSERT-PAGE-NUMBER.CS

Also, you use `[insertField]` API in Editor module to insert the Page number in current position

`typescript

//Current page number

```
container.documentEditor.editor.insertField('PAGE * MERGEFORMAT', '1');
```

`

Get page count

You can use `[pageCount]` API to gets the total number of pages in Document.

The following example code illustrates how to get the number of pages in Document.

CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        // To insert text in cursor position
        container.documentEditor.editor.insertText('Document editor');
        // To get the total number of pages
        var pageCount =container.documentEditor.pageCount;
    }
</script>
```

PAGECOUNT.CS

Navigate to specific page

You can use `[goToPage]` API in Selection module to move selection to the start of the specified page number.

The following example code illustrates how to move selection to specific page.

CSHTML

```

@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
    var documenteditor;
    var container;
    function onCreated() {
        var documenteditorElement = document.getElementById("container");
        container = documenteditorElement.ej2_instances[0];
        // To move selection to page number 2
        container.documentEditor.selection.goToPage(2);
    }
</script>

```

GO-TO-PAGE.CS

How to move the selection to specific position in Document Editor component

Using `select` API in selection module, You can set cursor position to anywhere in the document.

Selects content based on start and end hierarchical index

Hierarchical index will be in below format.

`sectionIndex;blockIndex;offset`

The following code snippet illustrate how to select using hierarchical index.

``typescript`

`// Selection will occur between provided start and end offset`

`this.documentEdContainerIns.documentEditor.editor.insertText("Welcome");`

`// The below code will select the letters "We" from inserted text "Welcome"`

`this.documentEdContainerIns.documentEditor.selection.select("0;0;0", "0;0;2");`

```

The following table illustrates about Hierarchical index in detail.

| Element | Hierarchical Format | Explanation |
|---------|---------------------|-------------|
|         | ----- ----- ----    |             |

| Move selection to Paragraph |sectionIndex;blockIndex;offset <br>**Ex:** 0;0;0| It moves the cursor to the start of paragraph. |

| Move selection to Table |sectionIndex;tableIndex;rowIndex;cellIndex;blockIndex;offset <br>**Ex:** 0;0;0;0;1;0| It moves the cursor to the second paragraph which is inside first row and cell of table. |

| Move selection to header |pageIndex;H;sectionIndex;blockIndex;offset<br>**Ex:** 1;H;0;0;0| It moves cursor to the header in second page. |

| Move selection to Footer |pageIndex;F;sectionIndex;blockIndex;offset<br>**Ex:** 1;F;0;0;0| It moves cursor to the footer in second page. |

#### *Get the selection start and end hierarchical index*

Using [startOffset], you can get start hierarchical index which denotes the start index of current selection.

Similarly, using [endOffset], you can get end hierarchical index which denotes the end index of current selection.

The following code snippet illustrate how to get the selection start and end offset on selection changes in document.

#### **CSHTML**

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
 var documenteditor;
 var container;
 function onCreated() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 documenteditor = container.documentEditor;
 // Event gets triggered on selection change in document
 container.selectionChange = function () {
 //Get the start index of current selection
 let startOffset = container.documentEditor.selection.startOffset;
 //Get the end index of current selection
 let endOffset = container.documentEditor.selection.endOffset;
 };
 }
</script>
```

#### **SELECT.CS**

#### *Selects the content based on left and top position*

Here, you can specify the [selection settings] to select the content based on left and top position.

x denotes the left position and y denotes the top position and extend denotes whether to extend or update selection.

Check below code sample for reference.

```
`typescript
```

```
this.container.documentEditor.selection.select({ x: 188.4814208984375 , y: 662.00005, extend: true });
```

```
,
```

How to disable header and footer edit in Document Editor component

*Disable header and footer edit in DocumentEditorContainer instance*

You can use `[restrictEditing]` property to disable header and footer editing based on selection context type.

RestrictEditing allows you to restrict the document modification and makes the Document read only mode. So, by using this property, and if selection inside header or footer, you can set this property as true.

The following example code illustrates how to header and footer edit in `DocumentEditorContainer` instance.

### CSHTML

```
<div>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").SelectionChange("onSelectionChanged").EnableToolbar(true).Render()
</div>
<script>
 var documenteditor;
 var container;
 function onCreated() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 }
 function onSelectionChanged() {
 // Check whether selection is in header
 if (container.documentEditor.selection.contextType.indexOf('Header')
> -1 ||
 // Check whether selection is in Footer
 container.documentEditor.selection.contextType.indexOf('Footer') >
-1) {
 // Change the document to read only mode
 container.restrictEditing = true;
 } else {
 // Change the document to editable mode
 container.restrictEditing = false;
 }
 }
</script>
```

### DOCUMENT-EDITOR.CS

Otherwise, you can disable clicking inside Header or Footer by using `[closeHeaderFooter]` API in selection module.



The following example code illustrates how to close header and footer when selection is inside header or footer in `DocumentEditorContainer` instance.

### CSHTML

```
<div>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").SelectionChange("onSelectionChanged").EnableToolbar(true).Render()
</div>
<script>
 var documenteditor;
 var container;
 function onCreated() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 }
 function onSelectionChanged() {
 // Check whether selection is in header
 if (container.documentEditor.selection.contextType.indexOf('Header')
> -1 ||
 // Check whether selection is in Footer
 container.documentEditor.selection.contextType.indexOf('Footer') >
-1) {
 // Close header Footer
 container.documentEditor.selection.closeHeaderFooter();
 }
 }
</script>
```

### DOCUMENT-EDITOR.CS

#### *Disable header and footer edit in DocumentEditor instance*

Like `restrictEditing`, you can use `[isReadOnly]` property in Document editor to disable header and footer edit.

The following example code illustrates how to header and footer edit in `DocumentEditor` instance.

### CSHTML

```
@Html.EJS().DocumentEditor("container").IsReadOnly(false).SelectionChange("onSelectionChanged").Render()
<script>
 var documentEditor;
 document.addEventListener('DOMContentLoaded', function () {
 documentEditor =
document.getElementById("container").ej2_instances[0];
 documentEditor.enableAllModules();
 });
 function onSelectionChanged() {
 // Check whether selection is in header
 if (documentEditor.selection.contextType.indexOf('Header') > -1 ||
 // Check whether selection is in Footer
 documentEditor.selection.contextType.indexOf('Footer') > -1) {
```

```

 // Change the document to read only mode
 documentEditor.isReadOnly = true;
 } else {
 // Change the document to editable mode
 documentEditor.isReadOnly = false;
 }
}
</script>

```

## DOCUMENT-EDITOR.CS

How to insert text, paragraph and rich-text content in Document Editor component

You can insert the text, paragraph and rich-text content in Document Editor component.

*Insert text in current cursor position*

You can use [insertText](#) API in editor module to insert the text in current cursor position.

The following example code illustrates how to add the text in current selection.

`typescript

// It will insert the provided text in current selection

```
this.container.documentEditor.editor.insertText('Syncfusion');
```

## CSHTML

```

<button id='insert'>Insert Text</button>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
 var documenteditor;
 var container;
 function onCreated() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 }
 document.getElementById('insert').addEventListener('click', () => {
 // It will insert the provided text in current selection
 container.documentEditor.editor.insertText('Syncfusion');
 });
</script>

```

## INSERT-TEXT.CS

*Insert paragraph in current cursor position*

To insert new paragraph at current selection, you can use `[insertText]` API with parameter as `\r\n` or `\n`.

The following example code illustrates how to add the new paragraph in current selection.

```
`typescript
// It will add the new paragraph in current selection
this.container.documentEditor.editor.insertText('\n');
`
```

#### *Insert the rich-text content*

To insert the HTML content, you have to convert the HTML content to SFDT Format using [\[web service\]](#). Then use [\[paste\]](#) API to insert the sfdt at current cursor position.

---

**Note:** Html string should be wellformatted html. [DocIO](#) support only wellformatted XHTML.

---

The following example illustrates how to insert the HTML content at current cursor position.

- Send the HTML content to server side for SFDT conversion. Refer to the following example to send the HTML content to server side and inserting it in current cursor position.

#### **CSHTML**

```
<button id='export'>Export</button>
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
 var documenteditor;
 var container;
 function onCreated() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 }
 let htmltags =
 "<?xml version='1.0' encoding='utf - 8'?><!DOCTYPE html PUBLIC '-
 //W3C//DTD XHTML 1.0 Strict//EN'http://www.w3.org/TR/xhtml1/DTD/xhtml1-
 strict.dtd'><html xmlns ='http://www.w3.org/1999/xhtml' xml:lang='en' lang
 ='en'><body><h1>The img element</h1><img
 src='https://www.w3schools.com/images/lamp.jpg' alt ='Lamp Image'
 width='500' height='600'/></body></html>";
 document.getElementById('export').addEventListener('click', () => {
 let http = new XMLHttpRequest();
 http.open('POST', '/api/documenteditor/LoadString');
 http.setRequestHeader('Content-Type', 'application/json;charset=UTF-8');
 http.responseType = 'json';
 http.onreadystatechange = function () {
 if (http.readyState === 4) {
 if (http.status === 200 || http.status === 304) {
 // Insert the sfdt content in cursor position using paste API
 container.documentEditor.editor.paste(http.response);
 } else {
 alert('failed;');
 }
 }
 };
 let htmlContent = { content: htmltags };
 http.send(JSON.stringify(htmlContent));
 });
}
```

```
});
</script>
```

### INSERT-RICH-TEXT.CS

- Refer the following code example for server-side web implementation for HTML conversion using DocumentEditor.

```
`c#
//API controller for the conversion.
[HttpPost]
public string LoadString([FromBody]InputParameter data)
{
 // You can also load HTML file/string from server side.
 Syncfusion.EJ2.DocumentEditor.WordDocument document =
 Syncfusion.EJ2.DocumentEditor.WordDocument.LoadString(data.content, FormatType.Html); // Convert
 the HTML to SFDT format.
 string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
 document.Dispose();
 return json;
}
public class InputParameter
{
 public string content {get; set; }
}
`
```

---

**Note:** The above example illustrates inserting HTML content. Similarly, you can insert any rich-text content by converting any of the supported file formats (DOCX, DOC, WordML, HTML, RTF) to SFDT.

---

#### [How to change the cursor color using CSS in Document Editor component](#)

Document Editor default cursor color is black. The user can change the color by overriding the css property using class name. The Document editor cursor css have a class named `e-de-blink-cursor`.

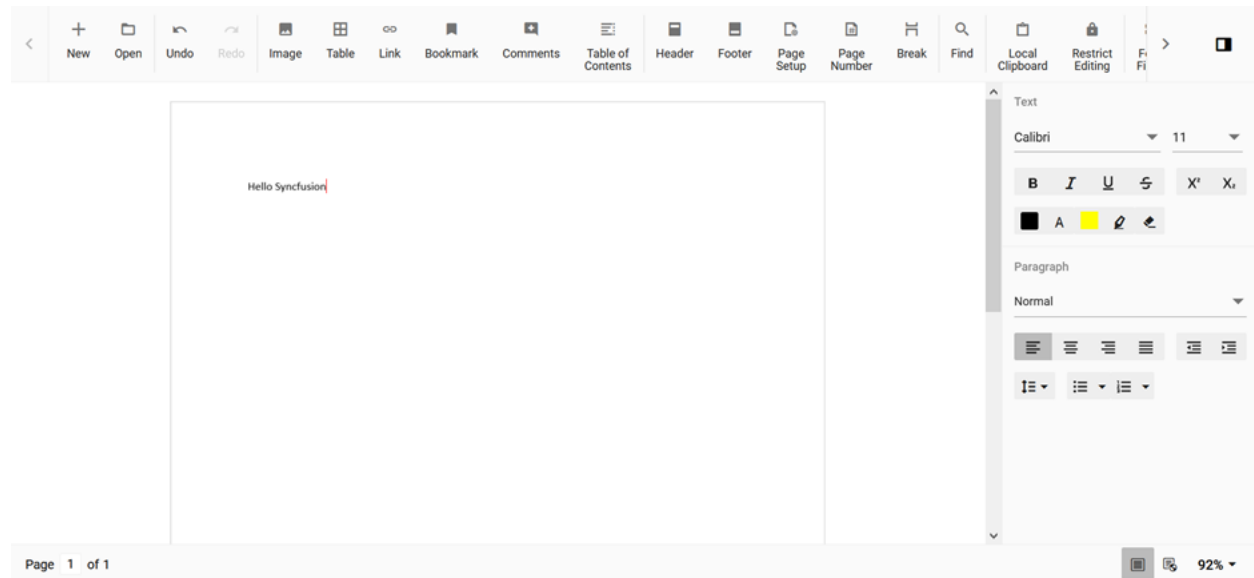
Refer the below code snippet to change the cursor color to red.

```
`css
.e-de-blink-cursor {
border-left: 1px solid red!important;
```

}

,

Output will be like below:



### How to hide the default tool bar and properties pane in Document Editor component

**Document editor container** provides the main document view area along with the built-in toolbar and properties pane.

**Document editor** provides just the main document view area. Here, the user can compose, view, and edit the Word documents. You may prefer to use this component when you want to design your own UI options for your application.

#### Hide the properties pane

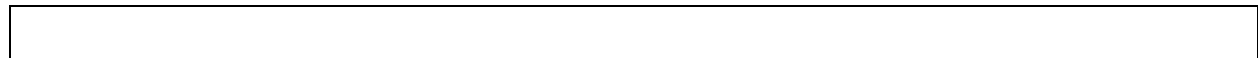
By default, Document editor container has built-in properties pane which contains options for formatting text, table, image and header and footer. You can use `[showPropertiesPane]` API in **DocumentEditorContainer** to hide the properties pane.

The following example code illustrates how to hide the properties pane.

#### CSHTML

```
@Html.EJS().DocumentEditorContainer("container").ShowPropertiesPane(false).EnableToolbar(true).Render()
```

#### HIDE-THE-DEFAULT-PROPERTIESPANE.CS



**Note:** Positioning and customizing the properties pane in Document editor container is not possible. Instead, you can hide the existing properties pane and create your own pane using public API's.

#### Hide the toolbar

You can use `[enableToolbar]` API in **DocumentEditorContainer** to hide the existing toolbar.

The following example code illustrates how to hide the existing toolbar.

### CSHTML

```
@Html.EJS().DocumentEditorContainer("container").EnableToolbar(false).Render()
```

### HIDE-THE-DEFAULT-TOOLBAR.CS

*See Also*

- [How to customize the toolbar](#)

How to insert text or image in table programmatically in Document Editor component

Using Document editor API's, you can insert [text] or [image] in [table] programmatically based on your requirement.

Use [selection] API's to navigate between rows and cells.

The following example illustrates how to create 2\*2 table and then add text and image programmatically.

### CSHTML

```
@Html.EJS().DocumentEditorContainer("container").Created("onCreated").Enable
Toolbar(true).Render()
<script>
 var documenteditor;
 var container;
 function onCreated() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 // To insert the table in cursor position
 container.documentEditor.editor.insertTable(2, 2);
 // To insert the image at table first cell

 container.documentEditor.editor.insertImage("data:image/png;base64,iVBORw0KG
 goAAAANSUHEUgAAAAUAAAFCAyAAACNbyblAAAAHE1EQVQI12P4
 //8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg==");
 // To move the cursor to next cell
 moveCursorToNextCell();
 // To insert the image at table second cell

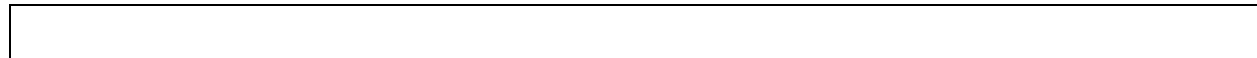
 container.documentEditor.editor.insertImage("data:image/png;base64,iVBORw0KG
 goAAAANSUHEUgAAAAUAAAFCAyAAACNbyblAAAAHE1EQVQI12P4
 //8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg==");
 // To move the cursor to next row
 moveCursorToNextRow();
 // To insert text in cursor position
 container.documentEditor.editor.insertText('Text');
 // To move the cursor to next cell
 moveCursorToNextCell();
 // To insert text in cursor position
```

```

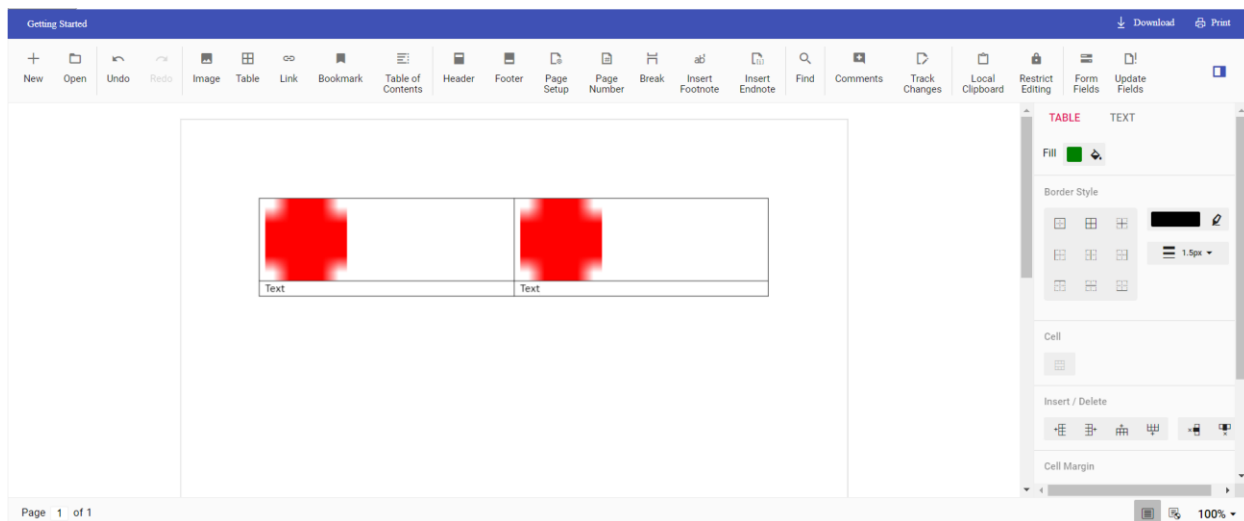
 container.documentEditor.editor.insertText('Text');
 }
 function moveCursorToNextCell() {
 // To get current selection start offset
 var startOffset = container.documentEditor.selection.startOffset;
 // Increasing cell index to consider next cell
 var startOffsetArray = startOffset.split(';');
 startOffsetArray[3] = parseInt(startOffsetArray[3]) + 1;
 // Changing start offset
 startOffset = startOffsetArray.join(';');
 // Navigating selection using select method
 container.documentEditor.selection.select(startOffset, startOffset);
 }
 function moveCursorToNextRow() {
 // To get current selection start offset
 var startOffset = container.documentEditor.selection.startOffset;
 // Increasing row index to consider next row
 var startOffsetArray = startOffset.split(';');
 startOffsetArray[2] = parseInt(startOffsetArray[2]) + 1;
 // Going back to first cell
 startOffsetArray[3] = 0;
 // Changing start offset
 startOffset = startOffsetArray.join(';');
 // Navigating selection using select method
 container.documentEditor.selection.select(startOffset, startOffset);
 }
}
</script>

```

## INSERT-TEXT-IMAGE-TABLE



The output will be like below.



How to change the default search highlight color in Document Editor component  
 Document editor provides an options to change the default search highlight color using  
 [searchHighlightColor] in Document editor settings. The highlight color which is given in

`documentEditorSettings` will be highlighted on the searched text. By default, search highlight color is yellow.

Similarly, you can use `[documentEditorSettings]` property for DocumentEditor also.

The following example code illustrates how to change the default search highlight color.

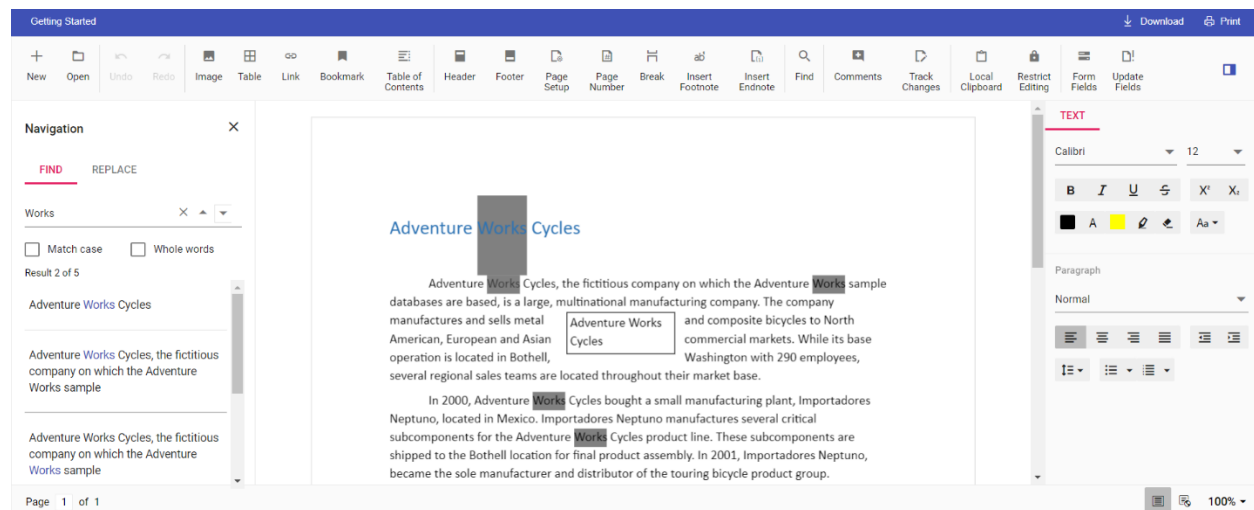
### CSHTML

```
@Html.EJS().DocumentEditorContainer("container").EnableToolbar(true).DocumentEditorSettings("settings").Render()

<script>
 var settings = { searchHighlightColor: 'Grey' };
</script>
```

### DOCUMENT-EDITOR.CS

Output will be like below:



### How to optimize the SFDT file

Starting from version v21.1.x, the SFDT file generated in Word Processor component is optimized by default to reduce the file size. All static keys are minified, and the final JSON string is compressed. This helps reduce the SFDT file size relative to a DOCX file and provides the following benefits,

- File transfer between client and server through the internet gets faster.
- The new optimized SFDT files require less storage space than the old SFDT files.

Hence, the optimized SFDT file can't be directly manipulated as JSON string.

This feature comes with a public API to switch between the old and new optimized SFDT format, allowing backward compatibility.

As a backward compatibility to create older format SFDT files, refer the following code changes,



Client/Server	Old Code	New Code from v21.1.x
Client-side	<code>{% elsif page.publishingplatform == "aspnet-mvc" %}{% include code-snippet/document-editor/optimize-sfdt/razorOld %}</code>	<code>{% elsif page.publishingplatform == "aspnet-mvc" %}{% include code-snippet/document-editor/optimize-sfdt/razor %}</code>
Server-side C#	<code>WordDocument sfdtDocument = WordDocument.Load(stream, formatType); string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(sfdtDocument);</code>	<code>WordDocument sfdtDocument = WordDocument.Load(stream, formatType); sfdtDocument.OptimizeSfdt = false; string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(sfdtDocument);</code>

To convert from older format SFDT from a new optimized SFDT file, refer the following code example,

Client/Server	Code example
Client-side	<code>{% elsif page.publishingplatform == "aspnet-mvc" %}{% include code-snippet/document-editor/optimize-sfdt/razor %}</code>
Server-side C#	<code>using(Syncfusion.DocIO.DLS.WordDocument docIODocument = WordDocument.Save(optimizedSfdt)) { sfdtDocument = WordDocument.Load(docIODocument); sfdtDocument.OptimizeSfdt = false; string oldSfdt = JsonSerializer.Serialize(sfdtDocument); }</code>

## Enable ruler

### [How to enable ruler in Document Editor component](#)

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor.

The following example illustrates how to enable ruler in Document Editor

### **CSHTML**

```
<div>
 @Html.EJS().Button("container_ruler_button").Content("Show/Hide
 Ruler").Click("onClick").Render()

 @Html.EJS().DocumentEditor("container").Created("onCreate").IsReadOnly(false)
 .DocumentEditorSettings("settings").Render()
</div>
<script>
 var container;
 var settings = { showRuler: true };
 function onCreate() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 container.enableAllModules();
 }
 function onClick() {
```

```
 container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
 }
</script>
```

### **DOCUMENT-EDITOR.CS**

```
public ActionResult Default()
{
 return View();
}
```

#### *How to enable ruler in Document Editor Container component*

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor Container.

The following example illustrates how to enable ruler in Document Editor Container.

### **CSHTML**

```
<div>
 @Html.EJS().Button("container_ruler_button").Content("Show/Hide
Ruler").Click("onClick").Render()

 @Html.EJS().DocumentEditorContainer("container").Created("onCreate").Document
EditorSettings("settings").Render()
</div>
<script>
 var container;
 var settings = { showRuler: true };
 function onCreate() {
 var documenteditorElement = document.getElementById("container");
 container = documenteditorElement.ej2_instances[0];
 }
 function onClick() {
 container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
 }
</script>
```

### **DOCUMENT-EDITOR.CS**

```
public ActionResult Default()
{
 return View();
}
```

## DropDownButton

### Getting Started with ASP.NET MVC DropDownButton Control

This section briefly explains about how to include [ASP.NET MVC DropDownButton](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

#### Add ASP.NET MVC DropDownButton control

Now, add the Syncfusion ASP.NET MVC DropDownButton control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@model List<object>
@Html.EJS().DropDownButton("element").Content("Edit").Items((IEnumerable<object>)Model).Render()
```

#### HOMECONTROLLER.CS

```
public ActionResult Index()
{
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 return View(items);
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DropDownButton control will be rendered in the default web browser.

Edit ▼

**Note:** [View Sample in GitHub.](#)

See also

- [DropDownButton with icons](#)
- [How to hide dropdown arrow](#)

## Icons and Styles in ASP.NET MVC DropDownButton Control

### DropDownButton icons

DropDownButton can have an icon to provide the visual representation of the action. To place the icon on a DropDownButton, set the [IconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the DropDownButton. You can customize the icon's position using the [IconPosition](#) property.

In the following example, the DropDownButton with default iconPosition and iconPosition as **Top** is showcased:

### CSHTML

```
@Html.EJS().DropDownButton("left-
icon").Content("Message").Items((IEnumerable<object>)ViewBag.items).IconCss(
"ddb-icons e-message").Render()
@Html.EJS().DropDownButton("top-
icon").Content("Message").Items((IEnumerable<object>)ViewBag.items).IconCss(
"ddb-icons e-
message").IconPosition(Syncfusion.EJ2.SplitButtons.SplitButtonIconPosition.T
op).Render()
<style>
/* csslint ignore:start */
@@font-face {
font-family: 'e-db-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXDNFudgAAABkAAAApnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAANmhoZWEHmQNTAAAArAAAACRobXR4D7gAAAAAYAAAAA
QBg9jYQB4ADoAAAHMAAAACmlheHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJCg9zdIwSr0
AAATcAAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAABAABAAAAAQAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAQACAAMAAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAEAAQAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAABAAKABwAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGCAAwABBAkAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAAwABBAk
```

```

ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLWljb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3RlZGlv3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAUADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGa
AAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtZWVpAtyZWf
kLXVucmVhZAZkZWxldGUAAAAAA==) format('truetype');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.ddb-icons {
 font-family: 'e-db-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-message::before {
 content: '\e703';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>

```

### ICONBUTTON.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult IconButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit"
 });
 items.Add(new
 {
 text = "Delete"
 });
 items.Add(new

```

```

 {
 text = "Mark as Read"
 });
 items.Add(new
 {
 text = "Like Message"
 });
 ViewBag.items = items;
 return View();
 }
}

```

**Note:** The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icons on the DropDownButton using the [IconCss](#) property.

### Vertical button

Vertical button in DropDownButton can be achieved by adding `e-vertical` class using [CssClass](#) property.

### CSHTML

```

@Html.EJS().DropDownButton("vertical-
btn").Content("Message").Items((IEnumerable<object>)ViewBag.items).IconCss("
ddb-icons e-message").Render()
<style>
 /* csslint ignore:start */
 @@font-face {
 font-family: 'e-db-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMj0jSRoAAAEoAAAVmNtYXNfudgAAABkAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAANmhoZWEHmQNTAAARAAAACRobXR4D7gAAAAAYAAAA
QbG9jYQB4ADoAAAHMAAAACmlheHABEAAAYAAABCAAAACBuYW1lH0mDAAAAPAAAAJcG9zdIwSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAACAAACAAAAAAAAAAAAEAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAA
AAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAQACAAMAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAzb+agICX/4+AcLXsv6cAWQBZAAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAEAAQAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAABAAKABWAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAACgAsADsAAQAAAAACwASAGcAAwABBAkAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLWlRiLWljb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMabwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIABlAHMAaQBuaGCAIABTAKAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGA
AAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbfWfPbAtyZWf
kLXVucmVhZAZkZWxldGUAAAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
 }

```

```
}
/* csslint ignore:stop */
.ddb-icons {
 font-family: 'e-db-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-message::before {
 content: '\e703';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>
```

#### VERTICALBUTTON.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult VerticalButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit"
 });
 items.Add(new
 {
 text = "Delete"
 });
 items.Add(new
 {
 text = "Mark as Read"
 });
 items.Add(new
 {
 text = "Like Message"
 });
 ViewBag.items = items;
 return View();
 }
 }
}
```



```
}

```

See Also

- [Dropdown popup with icons](#)
- [Customized icon size](#)

## Popup items

### Icons

The popup action item has an icon or image to provide visual representation of the action. To place the icon on a popup item, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the popup action item.

In the following sample, the icons for edit, delete, mark as read and like message menu items are added using the iconCss property.

### CSHTML

```
@Html.EJS().DropDownButton("left-
icon").Content("Message").Items((IEnumerable<object>)ViewBag.items).IconCss(
"ddb-icons e-message").Render()
<style>
 /* csslint ignore:start */
 @@font-face {
 font-family: 'e-db-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNfudgAAABkAAAApnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAANmhoZWEHmQNTAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQB4ADoAAAHMAAAACmlheHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJcG9zdIwSr0
AAATCAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQPUAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAQACAMAAAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAzb+agICX/4+AcLXsv6cAWQBZAAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAAAgAHAAsAAQAAAAAAAAAwAKABIAAQAAAAAAAAABAAKABwAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAACgAsADsAAQAAAAACwASAGcAAwABBAkAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLW1lLW1jb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMabwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIABlAHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGA
AAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbfWfPbAtyZWf
kLXVucmVhZAZkZWxldGUAAAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
 }
```

```
/* csslint ignore:stop */
.ddb-icons {
 font-family: 'e-db-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-message::before {
 content: '\e703';
}
.e-edit::before {
 content: '\ea9a';
}
.e-delete::before {
 content: '\e705';
}
.e-read::before {
 content: '\e704';
}
.e-like::before {
 content: '\e81a';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>
```

### POPUPICONS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult PopupIcons()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit",
 iconCss = "ddb-icons e-edit"
 });
 items.Add(new
 {
 text = "Delete",
 iconCss = "ddb-icons e-delete"
 });
 }
 }
}
```

```

 });
 items.Add(new
 {
 text = "Mark as Read",
 iconCss = "ddb-icons e-read"
 });
 items.Add(new
 {
 text = "Like Message",
 iconCss = "e-icons e-like"
 });
 ViewBag.items = items;
 return View();
}
}
}

```

### Navigations

Actions in DropDownButton can be used to navigate to the other web page when action item is clicked. This can be achieved by Popup items that can be customized using the [beforeItemRender](#) event. The item render event providing link to the action item using `url` property.

In the following sample, navigation URL for Flipkart, Amazon, and Snapdeal action items are added using the `url` property:

### CSHTML

```

@Html.EJS().DropDownButton("navigation").Content("Shop
By").Items((IEnumerable<object>)ViewBag.items).IconCss("e-cart-icon e-
link").BeforeItemRender("beforeItemRender").Render()
<script>
 function beforeItemRender(args) {
 args.element.getElementsByTagName('a')[0].setAttribute('target',
 '_blank');
 }
</script>
<style>
 /* csslint ignore:start */
 @@font-face {
 font-family: 'cart';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSQ4AAAEoAAAAVmNtYXNlEODVAAABiAAAADZnbHlmGat
ngwAAAcgAAADYaGVhZBktP4wAAADQAAAAANmhoZWEHmQNpAAAArAAAACRobXR4B+j//gAAAYAAAAA
IbG9jYQBsAAAAAHAAAAABmlheHABDwBQAAABCAAAACBuYW1lfiv21QAAAgAAAAIBcG9zdIZzcJA
AAASkAAAAOgABAAADUv9qAFoEAP/+//wD7AABAAAAAAAAAAAAAAAAAAAAgABAAAAAQAA2UwSaF8
PPPUACwPoAAAAANfSfWUAAAAA19J9Zf/+AAAD7APdAAAAACAACAAAAAAAAAAAAEAAAACAEQAawAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAQP0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPdAJYAAAAABAAAAAAAAABAAAAAPo//4
AAAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAEAAAA
AAAAAbAAAAAP//gAAA+wD3QAJABIAQwAAJR4BMjY0JicOAQUeATI2NCYiBgEOAwclIgyXEXx4BMwU
yFgcOAQclIgyXBhYXBT4BPwE2PwI2NxM+AycmIyIGAeABJzonJx0gJ/6jASc6Jyc6JwMXIDgZPyX
9giQkBkIIoyQBACQgCQo/JP7RIxcBARcjAVgkQg0QDgkICgkNkw4xMBwCBScJE1UdJyc6JwEBJx0
dJyc6JycDZQg0PigBAY0k/uAkMAQnHRsmAQMTDA8QAQMBLCilIhYWFxYhAYciNg4YDBMCAAAAEgD
eAAEAAAAAAAAAAAAEAAAAAAAAEABAABAAEAAAAAAAAIABwAFAAEAAAAAAAAAMABAAMAEAAAAAAAAQ
ABAAQAAEAAAAAAAAUACwAUAAEAAAAAAAAAYABAAfAAEAAAAAAAAoALAAjAAEAAAAAAAAsAEgBPAAMAAQ

```

```

JAAAAAgBhAAMAAQQJAAEACABjAAMAAQQJAAIADgBrAAMAAQQJAAACAB5AAMAAQQJAAQACACBAAM
AAQQJAAUAFgCJAAMAAQQJAAAYACACfAAMAAQQJAAoAWACnAAMAAQQJAAAsAJAD/IGNhcnRSZWdlbGF
yY2FydGNhcnRWZXJzaW9uIDEuMGNhcnRGb250IGdlbmVvYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1
ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAQAGMAYQByAHQAUGBlAGcAdQBsAGEAcgBjAGE
AcgB0AGMAYQByAHQAVgBlAHIAcwBpAG8AbgAgADEALgAwAGMAYQByAHQARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdABlAGQAaQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMAEHNob3BwaW5nLWNhcnQtMDUAAAA
A) format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.e-cart-icon {
 font-family: 'cart' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-link::before {
 content: '\e700';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>

```

## NAVIGATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult Navigation()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Flipkart",
 iconCss = "e-cart-icon e-link",
 url = "https://www.google.co.in/search?q=flipkart"
 });
 items.Add(new
 {
 text = "Amazon",

```

```

 iconCss = "e-cart-icon e-link",
 url = "https://www.google.co.in/search?q=amazon"
 });
 items.Add(new
 {
 text = "Snapdeal",
 iconCss = "e-cart-icon e-link",
 url = "https://www.google.co.in/search?q=snapdeal"
 });
 ViewBag.items = items;
 return View();
}
}
}

```

## Template

### Item templating

Popup items can be customized using the [beforeItemRender](#) event. The item render event triggers while rendering each popup action item. The event argument will be used to identify the action item and customize based on the requirement.

In the following example, the icons in each li items is right aligned by appending span element in li rendering:

### CSHTML

```

@Html.EJS().DropDownButton("item-
template").Items((IEnumerable<object>)ViewBag.items).IconCss("e-ddb-icons e-
paste").BeforeItemRender("beforeItemRender").Render()
<script>
 function beforeItemRender(args) {
 if (args.item.text === 'Edit') {
 args.element.innerHTML = '<div>Paste
Text<div>Provides option to paste only the
selected
text.</div></div>';
 args.element.style.height = '80px';
 var span = args.element.children[0];
 span.setAttribute('class', 'e-cm-icons e-pastetext e-align');
 var div = args.element.children[1];
 div.setAttribute('class', 'e-div-align');
 } else {
 args.element.innerHTML = '<div>Paste
Special<div>Provides options to paste formulas,
 values, comments,
validations etc...</div></div>';
 args.element.style.height = '80px';
 var span = args.element.children[0];
 span.setAttribute('class', 'e-cm-icons e-pastespecial e-align');
 var div = args.element.children[1];
 div.setAttribute('class', 'e-div-align');
 }
 }
</script>
<style>
 /* csslint ignore:start */
 @@font-face {

```

```
font-family: 'e-context-menu';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjJNRVMAAAEoAAAAVmNtYXDicOK6AAABjAAAADhnbHlmGcE
PFQAAAcwAAAMwaGVhZA69CA8AAADQAAAAANmhoZWEH9AQEAAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQDYAZgAAAHEAAAAACG1heHABEGDAAAABCAAAACBuYW11xY1dlQAABPwAAAKFcG9zdPJwcMo
AAAEeAAAAASAABAAAEAAAAAFwEAAAAAADlwABAAAAAAAAAAAAAAAAAAAAAAwABAAAAQAAGmhm8l8
PPPUACwQAAAAAANYD4Y8AAAAA1gPhjwAAAAADlwP0AAAAACAACAAAAAAAAAAAAEAAAADALQABQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQA ZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4mDiYQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAMAAAUAAMAAQAAABQABAakAAAABAAEAAEAAOJh//8AAOJg//8AAAAABAAQAAAA
BAAIAAAAAANgBmAFAAAAAAOXA/QABAALAC0ATgCzAAABISchJZcVHwc/By8HDwYBFSElMxEhESU
HFQ8GLwc/Bx8GJysBDw4RHw4zITM/DhEvDisBLw4rAQ8NAUQBdlxAfr0BAwQGBwgJCgkJCACGBAM
BAQMEBgcICQkKCAgHBgQD/qYB1179jQFoAQMEBgcHCQkJCQgGBgQDAQEDBAYGCAkJCQkHBwYEA6y
9CgkICQgHBwcGBQQAAMBAQEBAwMDBQUGBwcHCAkICQoCeAoJCAkIBwcHBgUEBAMDAQEBAQMDAwU
FBgCHBwgJCAkKvQqEBgUHBwcICQkJCgoKCwsLCwoKCgkJCQgHBwcFBgQBBYVRuh0FBQkIBwUFAgE
BAgUFBwgJCgkJCACGBAMBAQMEBgcICQEifX39LwLRMwQFCAGHBQUCAQECEBQUHCAgJCQkIBwUEAwE
BAwQFBwgJIgICAwQFBQYGBwgICakJCf0pCQkJCAGIBWYGBQUEAwICAgIDBAUFBgYHCAgICQkJAtc
JCQkICAGHBgYFBQQDAgIKCQkICAGHBgYFBAQDAgICAgMEBAUGBgICAgJCQAFAAAAAAAOXA/QABwA
PABcAOACdAAABHwIjPwIDMzcZfZMDIycVITUzESERJQCVDwYvBz8HHwYnKwEPDhEfDjMhMz8OES8
OKwEvDisBDw0B/wQKK3MmBQ6dMyeHKDWCO90B1179jQFoAQMEBgcHCQkJCQgGBgQDAQEDBAYGCAk
JCQkHBwYEA6y9CgkICQgHBwcGBQQAAMBAQEBAwMDBQUGBwcHCAkICQoCeAoJCAkIBwcHBgUEBAM
DAQEBAQMDAwUFBgCHBwgJCAkKvQqEBgUHBwcICQkJCgoKCwsLCwoKCgkJCQgHBwcFBgQCFREigG4
SM/6wd3cBe/t9ff0vAtEzBAUICACFBQIBAQIFBQcICakJCQgHBQQDAQEDBAUHCAkiAgIDBAUFBgY
HCAgICQkJ/SkJCQkICAGHBgYFBQQDAgICAgMEBQUGBgICAgJCQkC1wkJCQgICACGBgUFBAMCAgo
JCQgICACGBgUEBAMCAgICAwQEBQYGBwgICakJAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAA8
AAQABAAAAAAACAACAEAAABAAAAAADAA8AFwABAAAAAAAEAA8AJgABAAAAAAAFAA8ANQABAAAAAA
GAA8AQABAAAAAAAKACwATwABAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4AjwADAAE
ECQACAA4ArQADAAEECQADAB4AUwADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQAGAB4BDQA
DAEECQAKAFgBKwADAAEECQALACQBgyBDb250ZXh0TWVudSAoMilSZWdlbGFyQ29udGV4dE11bnU
gKDIpQ29udGV4dE11bnUgKDIpVmVyc2lubiAxLjBDb250ZXh0TWVudSAoMilGb250IGdlbmVyYXR
lZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAEMAbwB
uAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAFIAZQBnAHUAbABhAHIAQwBvAG4AdABlAHgAdABNAGU
AbgB1ACAAKAAyACkAQwBvAG4AdABlAHgAdABNAGUAbgB1ACAAKAAyACkAVgB1AHIAcWBPAG8AbgA
gADEALgAwAEMAbwBuAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAEYAbwBuAHQAIAABNAGUAbgB1AHIA
AYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAEQBuAGMAZgB1AHMAAQwBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgB1AHMAAQwBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAOoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQAD01UX1Bhc3RlU3B1Y2lhbAxNVF9QYXN
0ZVRleHQAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
/* csslint ignore:stop */
.e-cm-icons {
font-family: 'e-context-menu';
font-style: normal;
font-variant: normal;
font-weight: normal;
line-height: 1;
text-transform: none;
}
@@font-face {
font-family: 'ddb-icons';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjJ0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlm1lh3
3NQAAAdwAAAJMAgVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEWB1AAABCAAAACBuYW11LlBM9QAABCgAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAAABQABAAAAQAaojXaQl8
```

```

PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAEEAAAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAptAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAaaaaaWAAABQAaWABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAGADAAQAAAAAAI4AwgEAASYAAWAA//oDNQPsAA4AHQBYAAALHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFx06IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfikwVhcMBWylFRYkBWcMF1YwFCAALDAQUIxcUPhojDiAOuR4cAQvEwWsb6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsaKQ4ONzcniiYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAQAa+kABQANABcAHwAAARUzFSErAYERiZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+it8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVl+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRmxE
zETM1IQL+zPlabpADA5t0f2F+XP41afbmAZgBJwmYCHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAABAAAAQAAAAAAQAJAAEAQAAAAAAgAHAAoAAQAAAAAAAwAJABEAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADCAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHCAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBJAG8AbgBzAGQAZABiAC0AaQBJAG8AbgBzAFY
AZQByAHMAAQBVAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIAB1AHMAAQBuAGCAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3R1XzAxBGZvbnQ
OcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
 font-weight: normal;
 font-style: normal;
}
.e-ddb-icons {
 font-family: 'ddb-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-pastespecial::before {
 content: '\e260';
}
.e-pastetext::before {
 content: '\e261';
}
.e-paste::before {
 content: '\e701';
}
button {
 margin: 25px 5px 20px 20px;
}
.e-dropdown-popup ul {
 max-width: 400px;
 white-space: nowrap;

```

```

 }
 .e-align {
 float: left;
 width: 15%;
 margin-top: 15px;
 font-size: 45px;
 color: grey;
 }
 .e-div-align {
 float: right;
 width: 75%;
 line-height: 23px;
 margin: 0 15px 0 0;
 }
}
</style>

```

### ITEMTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult ItemTemplate()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit"
 });
 items.Add(new
 {
 text = "Cut"
 });
 ViewBag.items = items;
 return View();
 }
 }
}

```

### Popup Templating

The whole popup can be customized as per the requirement. In the following example, the popup can be customized by handling it in [target](#) property.

### CSHTML

```

<div id="target" style='border: 1px solid grey;'>
 <table class="e-popup-table">
 <caption class="e-caption" style='height: 18px; background-
color: #e0e0e0;'>Insert Table</caption>
 <tr class='e-row'>

```



```

 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
 <tr class='e-row'>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
 <tr class='e-row'>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
 <tr class='e-row'>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
 <tr class='e-row'>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
 <tr class='e-row'>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
 <tr class='e-row'>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 <td class='e-data'></td>
 </tr>
</table>
</div>

```

```

@Html.EJS().DropDownButton("element").Content("Table").Target("#target").IconCss("e-icons e-table").Render()

```

```

<style>
 .shortcut {
 float: right;
 margin-top: 9px;
 padding-left: 30px;
 }

```

```
 height: 100px;
 }
 .e-popup-table {
 border-collapse: separate;
 }
 .e-data {
 border: 1px solid grey;
 padding: 8px;
 }
 .e-table::before {
 content: '\e705';
 }
 .e-row {
 padding-left: 3px;
 padding-right: 3px;
 }
 .e-caption {
 caption-side: top;
 padding: 0 0 0 18px;
 }
 button {
 margin: 25px 5px 20px 20px;
 }
</style>
```

#### **POPUPTEMPLATE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult PopupTemplate()
 {
 return View();
 }
 }
}
```

#### See Also

- [Integration with ListView component](#)

#### Accessibility in Drop Down Button Component

The Drop down button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Drop down button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

#### WAI-ARIA attributes

The Drop down button component followed the [WAI-ARIA] patterns to meet the accessibility. The following ARIA attributes are used in the Drop down button component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Drop down button component as **button**, Drop down button popup as **menu**, and the dropdown popup action items as **menuitem**. |

| **aria-haspopup** | Indicates the availability of the popup element. |

| **aria-expanded** | Indicates whether the popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

### Keyboard interaction

The Dropdown button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Drop down button component.

| **Press** | **To do this** |

| --- | --- |

| **Esc** | Closes the popup. |

| **Enter** | Opens the popup, or activates the highlighted item and closes the popup. |

| **Space** | Opens the popup. |

| **Up** | Navigates up or to the previous action item. |

| **Alt + Up Arrow** | Closes the popup. |

| **Alt + Down Arrow** | Opens the popup. |

### Ensuring accessibility

The Drop down component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Drop down button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Drop down button component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

### Styles and Appearances

To modify the DropDownButton appearance, you need to override the default CSS of DropDownButton component. Find the list of CSS classes and its corresponding section in DropDownButton. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

| ----- | ----- |

| **.e-dropdown-btn** | To customize the dropdown button |

| **.e-dropdown-btn:hover** | To customize the dropdown button on hover |

- | .e-dropdown-btn.e-active | To customize the dropdown button on active |
- | .e-dropdown-popup | To customize the dropdown button pop up |
- | .e-dropdown-popup ul .e-item: hover | To customize the dropdown button pop up items on hover |
- | .e-dropdown-popup ul .e-item: active | To customize the dropdown button pop up items on active |

## How To

### Change caret icon

Dropdown arrow can be customized on popup open and close. It can be handled in [beforeOpen](#) and [beforeClose](#) event.

In the following example, the up arrow is updated on popup close and down arrow is updated on popup open using `beforeOpen` and `beforeClose` event by adding and removing `e-caret-up` class.

### CSHTML

```
@Html.EJS().DropDownButton("ddbtn").Content("Clipboard").Items((IEnumerable<
object>)ViewBag.items)
 .BeforeOpen("beforeOpen").BeforeClose("beforeClose").Render()
<style>
 .e-caret {
 transform: rotate(0deg);
 transition: transform 200ms ease-in-out;
 }
 .e-caret-up .e-caret {
 transform: rotate(180deg);
 }
</style>
<script>
 // Removing 'e-caret-up' class.
 function beforeClose (args) {
 this.cssClass = '';
 }
 // Adding 'e-caret-up' class.
 function beforeOpen (args) {
 this.cssClass = 'e-caret-up';
 }
</script>
```

### UPDOWN.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult HideArrow()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
```

```

 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
}
}

```

### Create DropDownButton with rounded corner

DropDownButton with rounded corner can be achieved by adding `border-radius` CSS property to button element.

In the following example, `e-round-corner` class is defined with `5px border-radius` property and added that class to button element using `cssClass` property.

#### CSHTML

```

@Html.EJS().DropDownButton("ddbtn").Content("Clipboard").Items((IEnumerable<
object>)ViewBag.items).CssClass("e-round-corner").Render()
<style>
 .e-round-corner {
 border-radius: 5px;
 }
 button {
 margin: 25px 5px 20px 20px;
 }
</style>

```

#### ROUND.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Buttons;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult DropDownButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 }
 }
}

```

```

 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
}

```

### Create right-to-left DropDownButton

DropDownButton component has RTL support. This can be achieved by setting [enableRtl](#) as true.

### CSHTML

```

@Html.EJS().DropDownButton("rtl").Content("Message").Items((IEnumerable<obje
ct>)ViewBag.items).IconCss("ddb-icons e-message").EnableRtl(true).Render()
<style>
/* csslint ignore:start */
@@font-face {
 font-family: 'e-db-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0jSROAAAEoAAAAVmNtYXNfudgAAABkAAAApnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAAAANmhoZWEHmQNTAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJcG9zdIwSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAAAAABAABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAAAAAEAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAwGPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAQACAMAAAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAAAAQAKAAEAAQAAAAAAAAAgAHAAsAAQAAAAAAAAAwAKABIAAQAAAAAAAAABAAKABwAAQAAAAAABQALACY
AAQAAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGCAAwABBAKAAAACAHkAAwABBAKAAQA
UAHsAAwABBAKAAgAOAI8AAwABBAKAAwAUAJ0AAwABBAKABAAUALEAAwABBAKABQAWAMUAAwABBAK
ABgAUANsAAwABBAKACgBYAO8AAwABBAKACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLW1lLW1jb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIABlAHMAaQBwAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGA
AAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbfWfPbAtyZWf
kLXVucmVhZAZkZWxldGUAAAAAAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.ddb-icons {

```

```
font-family: 'e-db-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-message::before {
 content: '\e703';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>
```

## RTL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult Rtl()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit"
 });
 items.Add(new
 {
 text = "Delete"
 });
 items.Add(new
 {
 text = "Mark as Read"
 });
 items.Add(new
 {
 text = "Like Message"
 });
 ViewBag.items = items;
 return View();
 }
 }
}
```



### Customize icon and width

Width of the DropDownButton can be customized by setting required width to the dropdown element.

The following UI can be achieved by setting [iconPosition](#) as **Top**, width as **85px** and size of the font icon as **40px** by adding **e-custom** class.

#### CSHTML

```
@Html.EJS().DropDownButton("ddbtn").Content("Clipboard").Items((IEnumerable<object>)ViewBag.items).BeforeItemRender("itemRender").Render()
<style>
 button {
 margin: 25px 5px 20px 20px;
 }
</style>
<script>
function itemRender(args) {
 if (args.item.text === 'Copy') {
 //To underline a particular text.
 args.element.innerHTML = '<u>C</u>opy';
 }
}
</script>
```

#### UNDERLINE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Buttons;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult DropDownButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
 }
}
```

## Disable a DropDownButton

DropDownButton component can be enabled or disabled by giving [disabled](#) property. To disable DropDownButton component, the disabled property can be set as `true`.

### CSHTML

```
@Html.EJS().DropDownButton("disabled").Content("Message").Items((IEnumerable<object>)ViewBag.items).IconCss("ddb-icons e-message").Render()
@Html.EJS().Button("onDisable").CssClass("e-primary").Content("Disable").Render()
<script>
 document.getElementById('onDisable').onclick = function () {
 var dropdownObj =
ej.base.getInstance(document.getElementById('disabled'),
ej.splitbuttons.DropDownButton);
 dropdownObj.disabled = true;
 }
</script>
<style>
/* csslint ignore:start */
@@font-face {
 font-family: 'e-db-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAVmNtYXNfudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBKtK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJCg9zdIwSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAACAACAAAAAAAAAAAAEAAAEAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPUAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAjgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAQACAAMAAAAAAAAAHAA6AFwAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQIma6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAABAAKABWAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGcAAwABBAkAAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLWwRiLWljb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQBYAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUACgBzAGkABwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8ABgB0ACAAZwB1AG4
AZQBYAGEAdAB1AGQAIAB1AHMAaQBwAGcAIABTAHkABgBjAGYAdQBzAGkABwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkABgBjAGYAdQBzAGkABwBuAC4AYwBvAG0AAAAAAGa
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbfWfPbAtyZWf
kLXVucmVhZAZkZWxldGUAAAAAA==) format('truetype');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.ddb-icons {
 font-family: 'e-db-icons' !important;
 speak: none;
```

```
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-message::before {
 content: '\e703';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>
```

### **DISABLED.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult DEisable()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit"
 });
 items.Add(new
 {
 text = "Delete"
 });
 items.Add(new
 {
 text = "Mark as Read"
 });
 items.Add(new
 {
 text = "Like Message"
 });
 ViewBag.items = items;
 return View();
 }
 }
}
```

### Group popup items with ListView component

Header in popup items is possible in DropDownButton by templating entire popup with ListView. Create ListView with id `#listview` and provide it as a [target](#) for DropDownButton.

In the following example, ListView element is given as `target` to DropDownButton and header can be achieved by [groupBy](#) property.

#### CSHTML

```
@using Syncfusion.EJ2.Lists;
<div>

@Html.EJS().ListView("listview").Enable(true).DataSource((IEnumerable<object>)ViewBag.listdata).ShowCheckBox(true).Fields(new ListViewFieldSettings {
 GroupBy = "category", Text = "text" }).Render()
 @Html.EJS().DropDownButton("ddbtn").CssClass("e-caret-hide").IconCss("e-icons e-down").Target("#listview").Render()
</div>
<style>
 .e-down::before {
 content: '\e969';
 }
 #listview {
 display: block;
 max-width: 600px;
 margin: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
 }
 button {
 margin: 30% 5px 20px 30%;
 }
</style>
```

#### LISTVIEW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult DropDownButton()
 {
 List<object> listdata = new List<object>();
 listdata.Add(new
 {
 text = "Print",
 id = "data1",
 category = "Customize Quick Access Toolbar"
 });
 listdata.Add(new
 {
```

```

 text = "Save As",
 id = "data2",
 category = "Customize Quick Access Toolbar"
 });
 listdata.Add(new
 {
 text = "Update Folder",
 id = "data3",
 category = "Customize Quick Access Toolbar"
 });
 listdata.Add(new
 {
 text = "Reply",
 id = "data4",
 category = "Customize Quick Access Toolbar"
 });
 ViewBag.listdata = listdata;
 return View();
}
}
}

```

### Hide dropdown arrow

You can hide the dropdown arrow from the DropDownButton by adding class `e-caret-hide` to DropDownButton element using [cssClass](#) property.

### CSHTML

```

@Html.EJS().DropDownButton("hide-
arrow").Content("Edit").Items((IEnumerable<object>)ViewBag.items).CssClass("
e-caret-hide").Render()

```

### HIDEARROW.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class DropDownButtonController : Controller
 {
 public ActionResult HideArrow()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new

```

```

 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
}
}

```

### Open a dialog on popup item click

This section explains about how to open a dialog on DropdownButton popup item click. This can be achieved by handling dialog open in [select](#) event of the DropdownButton.

In the following example, Dialog will open while selecting **Other Folder...** item.

### CSHTML

```

<div>
 @Html.EJS().Dialog("dialog").Header("Move Items").Content("Move Items To
 'Web Team'").Width("250px").Visible(false).Position(obj =>
 obj.X("100").Y("100")).Buttons(ViewBag.button).Render()

 @Html.EJS().DropDownButton("ddbtn").Content("Move").Items((IEnumerable<objec
 t>)ViewBag.items).CssClass("e-vertical").IconCss("ddb-icons e-
 folder").IconPosition(Syncfusion.EJ2.SplitButtons.SplitButtonIconPosition.To
 p).Select("onSelect").Render()
</div>
<style>
 /* csslint ignore:start */
 @@font-face {
 font-family: 'e-db-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-
 8;base64,AAEAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNfudgAAABkAAAAADpnbHlmSrK
 TCAAAAdgAAAC4aGVhZBktK8cAAADQAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
 QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJcG9zdIwksr0
 AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
 PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAACAACAAAAAAAAAAAAEAAAAEAAwAAgAAAA
 AAgAAAAoACgAAAP8AAAAAAAAAAQPUAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
 AAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
 D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABAAADnBf//AADnA//AAAAQA
 EAAAAQACAAMAAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
 NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
 EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zC
 BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
 AAAAAQAKAAEAQAAAAAAAAgAHAAsAAQAAAAAAAAAwAKABIAAQAAAAAAAABAABAAQAAAAAAAAABQALAC
 YAAQAAAAAAAAABgAKADEAAQAAAAACgAsADsAAQAAAAACwASAGCAAwABBAkAAAACAHkAAwABBAkAAQA
 UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
 ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAKAUcgZS1kYilpY29uc1JlZ3VsYXJlLW1lLW1lLW1l
 zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0Zvb250ZS1kYilpY29uc0Zvb250ZS1kYil
 pY29uc1JlZ3VsYXJlLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1lLW1l
 mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
 AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMAbwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
 WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
 AZQByAGEAdABlAGQAIABlAHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAAGB
 vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGa

```

```

AAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbWFpbAtyZWFKLXVucmVhZAZkZWxldGUAAAAAAAAA==) format('truetype');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.ddb-icons {
 font-family: 'e-db-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-folder::before {
 content: '\e703';
}
button {
 margin: 25px 5px 20px 20px;
}
</style>
<script>
 function onSelect(args){
 if (args.item.text === 'Other Folder...') {
 ej.base.getInstance(document.getElementById('dialog'),
 ejs.popups.Dialog).show();
 }
 }
 function btnClick () {
 ej.base.getInstance(document.getElementById('dialog'),
 ejs.popups.Dialog).hide();
 }
</script>

```

### DIALOGBUTTON-CORE.CS

```

public ActionResult DialogButton()
{
 ViewBag.DialogButtons = new {
 isPrimary = true,
 cssClass = "e-flat",
 content = "Submit",
 click = "dlgButtonClick"
 };
 return View();
}

```

### Position popup open

Popup open position can be changed according to the requirement. Popup open position can be changed in [open](#) event by setting `top` and `left` for the popup element.

In the following example, the **top** position of the popup element is changed in **open** event.

### CSHTML

```
@Html.EJS().DropDownButton("ddbtn").Content("Clipboard").Items((IEnumerable<object>)ViewBag.items).CssClass("e-caret-up")
 .Open("onOpen").Render()
<style>
 button {
 margin: 25% 5px 20px 30%;
 }
</style>
<script>
 function onOpen(args) {
 args.element.parentElement.style.top =
ej.base.getInstance(document.getElementById('ddbtn'),
ej.splitbuttons.DropDownButton).element.getBoundingClientRect().top -
args.element.parentElement.offsetHeight + 'px';
 }
</script>
```

### POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Buttons;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult DropDownButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
 }
}
```



### Underline a character in the item text

Underline a particular character in a text can be handled in [beforeItemRender](#) event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

In the following example, **C** is underlined in the text **Copy**.

#### CSHTML

```
@Html.EJS().DropDownButton("ddbtn").Content("Clipboard").Items((IEnumerable<object>)ViewBag.items).BeforeItemRender("itemRender").Render()
<style>
 button {
 margin: 25px 5px 20px 20px;
 }
</style>
<script>
function itemRender(args) {
 if (args.item.text === 'Copy') {
 //To underline a particular text.
 args.element.innerHTML = '<u>C</u>opy';
 }
}
</script>
```

#### UNDERLINE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Buttons;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult DropDownButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
 }
}
```

## DropDownList

### Getting Started with ASP.NET MVC DropDownList Control

This section briefly explains about how to include [ASP.NET MVC DropDownList](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
```

```
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC DropDownList control

Now, add the Syncfusion ASP.NET MVC DropDownList control in `~/Views/Home/Index.cshtml` page.

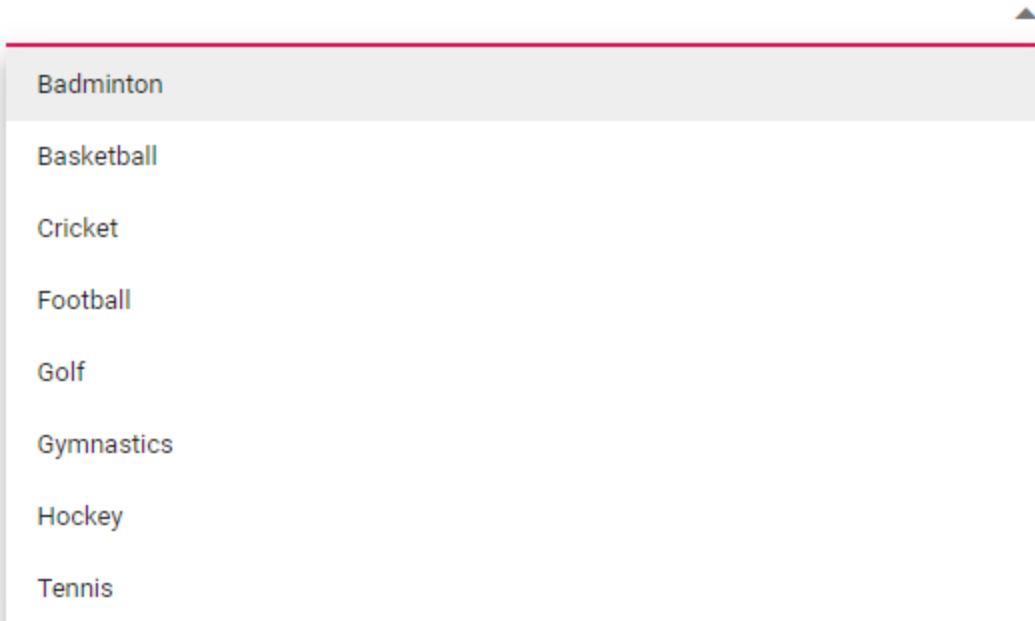
#### CSHTML

```
@model List<string>
@Html.EJS().DropDownList("games").DataSource((IEnumerable<object>)Model).Render()
```

#### HOMECONTROLLER.CS

```
public ActionResult Index()
{
 List<string> data = new List<string>() { "Badminton", "Basketball",
 "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View(data);
}
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DropDownList control will be rendered in the default web browser.



### Binding data source

After initialization, populate the DropDownList with data using the [dataSource](#) property. Here, an array of string values is passed to the DropDownList control.

The following example illustrates the output in your browser.

#### CSHTML

```
@model List<string>
@Html.EJS().DropDownList("games").Placeholder("Select a
game").DataSource((IEnumerable<object>)Model).Render()
```

#### HOMECONTROLLER.CS

```
public ActionResult Index()
{
 List<string> data = new List<string>() { "Badminton", "Basketball",
 "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View(data);
}
```

### Configure the popup list

By default, the width of the popup list automatically adjusts according to the DropDownList input element's width, and the height of the popup list has '300px'.

The height and width of the popup list can also be customized using the [popupHeight](#) and [popupWidth](#) properties respectively.

In the following sample, popup list's width and height are configured.

#### CSHTML

```
@model List<string>
```

```
@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").PopupWidth("300px").DataSource((IEnumerable<obje
ct>)Model).Render()
```

## HOMECONTROLLER.CS

```
public ActionResult Index()
{
 List<string> data = new List<string>() { "Badminton", "Basketball",
 "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View(data);
}
```

**Note:** [View Sample in GitHub.](#)

See also

- [How to bind the data](#)

## Data Binding

The DropDownList loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of [array](#) or [DataManager](#).

The DropDownList also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of [DataManager](#) adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

**Note:** When binding complex data to the DropDownList, fields should be mapped correctly. Otherwise, the selected item remains undefined.

## Binding local data

Local data can be represented in two ways as described below.

### 1. Array of simple data

The DropDownList has support to load array of primitive data such as strings and numbers. Here, both value and text field acts the same.

## CSHTML

```
@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Re
nder()
```

**ARRAYOFSTRINGS.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult arrayofstrings()
 {
 ViewBag.data = new string[] { "Badminton", "Basketball",
"Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View();
 }
 }
}

```

*2. Array of JSON data*

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Vegetable** column from complex data have been mapped to the **value** field.

**CSHTML**

```

@Html.EJS().DropDownList("vegetables").Placeholder("Select a
Vegetables").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.da
ta).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value =
"Vegetable" }).Render()

```

**VEGETABLES.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Vegetables
 {
 public string Vegetable { get; set; }
 public string Category { get; set; }
 public string Id { get; set; }
 public List<Vegetables> VegetablesList()
 {
 List<Vegetables> veg = new List<Vegetables>();
 veg.Add(new Vegetables { Vegetable = "Cabbage", Category= "Leafy and
Salad", Id= "item1" });
 veg.Add(new Vegetables { Vegetable = "Chickpea", Category= "Beans",
Id= "item2" });
 veg.Add(new Vegetables { Vegetable = "Garlic", Category= "Bulb and
Stem", Id= "item3" });
 }
 }
}

```

```

 veg.Add(new Vegetables { Vegetable = "Green bean", Category=
"Beans", Id= "item4" });
 veg.Add(new Vegetables { Vegetable = "Horse gram", Category=
"Beans", Id= "item5" });
 veg.Add(new Vegetables { Vegetable = "Nopal", Category= "Bulb and
Stem", Id= "item6" });
 veg.Add(new Vegetables { Vegetable = "Onion", Category= "Bulb and
Stem", Id= "item7" });
 veg.Add(new Vegetables { Vegetable = "Pumpkins", Category= "Leafy
and Salad", Id= "item8" });
 veg.Add(new Vegetables { Vegetable = "Spinach", Category= "Leafy and
Salad", Id= "item9" });
 veg.Add(new Vegetables { Vegetable = "Wheat grass", Category= "Leafy
and Salad", Id= "item10" });
 veg.Add(new Vegetables { Vegetable = "Yarrow", Category = "Leafy and
Salad", Id = "item11" });
 return veg;
 }
}

```

### 3. Array of Complex data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Code.Id` column and `Country.CountryId` column from complex data have been mapped to the `value` field and `text` field, respectively.

#### CSHTML

```

@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<Object>)ViewBag.data).Fi
elds(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text =
"Country.CountryId", Value = "Code.Id" }).Render()

```

#### COMPLEX.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Code
 {
 public string Id { get; set; }
 }
 public class Country
 {
 public string CountryId { get; set; }
 }
 public class Complex
 {
 public Country Country { get; set; }
 public Code Code { get; set; }
 }
}

```

```

public List<Complex> GetData()
{
 List<Complex> data = new List<Complex>();
 data.Add(new Complex() { Country = new Country() { CountryId =
"Australia" }, Code = new Code() { Id = "AU" } });
 data.Add(new Complex() { Country = new Country() { CountryId =
"Bermuda" }, Code = new Code() { Id = "BM" } });
 data.Add(new Complex() { Country = new Country() { CountryId =
"Canada" }, Code = new Code() { Id = "CA" } });
 data.Add(new Complex() { Country = new Country() { CountryId =
"Cameroon" }, Code = new Code() { Id = "CM" } });
 data.Add(new Complex() { Country = new Country() { CountryId =
"Denmark" }, Code = new Code() { Id = "DK" } });
 data.Add(new Complex() { Country = new Country() { CountryId =
"France" }, Code = new Code() { Id = "FR" } });
 return data;
}
}

```

### Binding remote data

The DropDownList supports retrieval of data from remote data services with the help of [DataManager](#) control. The [Query](#) property is used to fetch data from the database and bind it to the DropDownList.

The following sample displays the first 6 contacts from **Customers** table of the **Northwind** Data Service.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Text = "ContactName",
 Value = "CustomerID"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()

```

### REMOTEDATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult remotedata()
 {
 return View();
 }
 }
}

```



```

 }
}
}

```

### *Bind to URL Adaptor*

The DropDownList supports retrieval of data from URL adaptor.

### **CSHTML**

```

@Html.EJS().DropDownList("games").Placeholder("Select a
Country").PopUpHeight("200px").DataSource(dataManger =>

dataManger.Url("/ComboBox/UrlDatasource").Adaptor("UrlAdaptor").CrossDomain(
true).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "ShipCountry"
}).Render()

```

### **URLDATA.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropdownlistController : Controller
 {
 public ActionResult Index()
 {
 return View();
 }
 public ActionResult UrlDatasource([FromBody]Data dm)
 {
 var val = OrdersDetails.GetAllRecords();
 var Data = val.ToList();
 var count = val.Count();
 if (dm.where != null)
 {
 Data = (from cust in Data
 where
cust.ShipCountry.ToLower().StartsWith(dm.@where[0].value.ToString())
 select cust).ToList();
 }
 if (dm.take != 0)
 Data = Data.Take(dm.take).ToList();
 return Json(Data);
 }
 public class Data
 {
 public int take { get; set; }
 public List<Wheres> where { get; set; }
 }
 public class Wheres

```

```

 {
 public string field { get; set; }
 public bool ignoreAccent { get; set; }
 public bool ignoreCase { get; set; }
 public bool isComplex { get; set; }
 public string value { get; set; }
 public string Operator { get; set; }
 }
 public class OrdersDetails
 {
 public static List<OrdersDetails> order = new
List<OrdersDetails>();
 public OrdersDetails()
 {
 }
 public OrdersDetails(int OrderID, string CustomerId, int
EmployeeId, double Freight, bool Verified, DateTime OrderDate, string
ShipCity, string ShipName, string ShipCountry, DateTime ShippedDate, string
ShipAddress)
 {
 this.OrderID = OrderID;
 this.CustomerID = CustomerId;
 this.EmployeeID = EmployeeId;
 this.Freight = Freight;
 this.ShipCity = ShipCity;
 this.Verified = Verified;
 this.OrderDate = OrderDate;
 this.ShipName = ShipName;
 this.ShipCountry = ShipCountry;
 this.ShippedDate = ShippedDate;
 this.ShipAddress = ShipAddress;
 }
 public static List<OrdersDetails> GetAllRecords()
 {
 if (order.Count() == 0)
 {
 int code = 10000;
 for (int i = 1; i < 10; i++)
 {
 order.Add(new OrdersDetails(code + 1, "ALFKI", i +
0, 2.3 * i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro",
"Denmark", new DateTime(1996, 7, 16), "Kirchgasse 6"));
 order.Add(new OrdersDetails(code + 2, "ANATR", i +
2, 3.3 * i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha",
"Brazil", new DateTime(1996, 9, 11), "Avda. Azteca 123"));
 order.Add(new OrdersDetails(code + 3, "ANTON", i +
1, 4.3 * i, true, new DateTime(1957, 11, 30), "Cholchester",
"Frankenversand", "Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave.
Bolívar #65-98 Llano Largo"));
 order.Add(new OrdersDetails(code + 4, "BLONP", i +
3, 5.3 * i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel",
"Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
 order.Add(new OrdersDetails(code + 5, "BOLID", i +
4, 6.3 * i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes",
"Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
 code += 5;
 }
 }
 }
 }

```

```

 }
 return order;
}
public int? OrderID { get; set; }
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
public bool Verified { get; set; }
public DateTime OrderDate { get; set; }
public string ShipName { get; set; }
public string ShipCountry { get; set; }
public DateTime ShippedDate { get; set; }
public string ShipAddress { get; set; }
}
}
}

```

### Web API Adaptor

Use the **WebApiAdaptor** to bind DropDownList with Web API created using OData.

### CSHTML

```

<div id='web-data' class='col-lg-6' style='padding-top:15px'>
 <div class='content'>
 @Html.EJS().DropDownList("games").DataSource(dataManger =>
dataManger.Url("/api/Dropdownlist/").Adaptor("WebApiAdaptor")).Render()
 </div>
</div>

```

### WEBAPI.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 [Route("api/[controller]")]
 public class DropdownlistController : Controller
 {
 List<string> game = new List<string>();
 [HttpGet]
 public List<string> Get()
 {
 game.Add("Badminton");
 game.Add("Basketball");
 game.Add("Cricket");
 game.Add("Golf");
 game.Add("Gymnastics");
 game.Add("Tennis");
 game.Add("Hockey");
 return game;
 }
 }
}

```

```

 }
}
}

```

### Binding with OData services

**OData** is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the **DataManger**.

### CSHTML

```

<div id='o-data' class='col-lg-6' style='padding-top:15px'>
 <div class='content'>
 @Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Order
s/").Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
 {
 Value = "CustomerID"
 }).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
 </div>
</div>

```

### ODATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropdownlistController : Controller
 {
 public ActionResult odata()
 {
 return View();
 }
 }
}

```

### Offline mode

To avoid post back for every action, set the DropDownList to load all data on initialization and make the actions process in client-side. To enable this behavior, use the **Offline** property of **DataManager**.

### CSHTML

```

<div id='o-data' class='col-lg-6' style='padding-top:15px'>
 <div class='content'>
 @Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

```

```
dataManger.Url("http://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders/").Offline(true).Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "CustomerID"
}).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
</div>
</div>
```

### OFFLINE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropdownlistController : Controller
 {
 public ActionResult offline()
 {
 return View();
 }
 }
}
```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)
- [How to get the count of the data when using remote data](#)
- [How to acheive cascading](#)
- [How to add item in between the options](#)
- [How to remove an item](#)
- [How to preselect the items in dropdownlist](#)

### Templates

The DropDownList has been provided with several options to customize each list item, group title, selected value, header, and footer elements. It uses the Essential JS 2 **Template engine** to compile and render the elements properly.

#### Item template

The content of each list item within the DropDownList can be customized with the help of [itemTemplate](#) property.

### CSHTML

```
@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("O
DataV4Adaptor").CrossDomain(true)).ItemTemplate("${FirstName}<span class
='city'>${City}").Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "FirstName"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
.city {
 right: 15px;
 position: absolute;
}
</style>
```

### ITEMTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult Itemtemplate()
 {
 return View();
 }
 }
}
```

### Value template

The currently selected value that is displayed by default on the DropDownList input element can be customized using the [valueTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the DropDownList input, which is separated by a hyphen.

### CSHTML

```
@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("${FirstName}<span class
='city'>${City}").ValueTemplate("${FirstName} -
```

```

${City}").Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
 {
 Value = "FirstName",
 }).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
 .city {
 right: 15px;
 position: absolute;
 }
</style>

```

### VALUETEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult valuetemplate()
 {
 return View();
 }
 }
}

```

### Group template

The group header title under which appropriate sub-items are categorized can also be customized with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).GroupTemplate("${City}</st
rong>").Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
 {
 Value = "FirstName",
 GroupBy = "City"
 }).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6).where(new ej.data.Predicate('City',
'equal','london').or('City','equal','seattle'))").Render()

```

**GROUPTEMPLATE.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult grouptemplate()
 {
 return View();
 }
 }
}

```

## Header template

The header element is shown statically at the top of the popup list items within the DropDownList, and any custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

**CSHTML**

```

@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("<span class='item'
>${FirstName}${City}").HeaderTemplate("NameCity").Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "FirstName",
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
 .head, .item {
 display: table;
 width: 100%;
 margin: auto;
 }
 .head {
 height: 40px;
 font-size: 15px;
 font-weight: 600;
 }
 .name, .city {
 display: table-cell;

```



```

 vertical-align: middle;
 width: 50%;
 }
 .head .name {
 text-indent: 16px;
 }
 .head .city {
 text-indent: 10px;
 }
</style>

```

### HEADERTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult headertemplate()
 {
 return View();
 }
 }
}

```

### Footer template

The DropDownList has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the DropDownList.

### CSHTML

```

@Html.EJS().DropDownList("games").DataSource((IEnumerable<object>)ViewBag.data).Placeholder("Select a game").PopupHeight("270px").FooterTemplate(" Total list items: " + 5 + "").Render()
<style>
 .foot {
 text-indent: 1.2em;
 display: block;
 font-size: 15px;
 line-height: 40px;
 border-top: 1px solid #e0e0e0;
 }
</style>

```

### FOOTERTEMPLATE.CS

```

using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult footertemplate()
 {
 ViewBag.data = new string[] { "Badminton", "Basketball",
 "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View();
 }
 }
}

```

### No records template

The DropDownList is provided with support to custom design the popup list content when no data is found and no matches are found on search with the help of [noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

### CSHTML

```

@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).No
RecordsTemplate(" NO DATA AVAILABLE").Render()

```

### NORECORDSTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult norecords()
 {
 ViewBag.data = new string[] { };
 return View();
 }
 }
}

```

### Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the DropDownList displays the notification.

## CSHTML

```
@Html.EJS().DropDownList("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/") .Adaptor("ODataV4Adaptor").CrossDomain(true).ActionFailureTemplate(" Data fetch get fails").Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "FirstName"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName']).take(6)").Render()
```

## ACTIONFAILURETEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult actionfailure()
 {
 return View();
 }
 }
}
```

See Also

- [How to acheive filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)
- [How to render tooltip for the options](#)

## Virtualization in DropDown List

Dropdown list virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a dropdown list activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the DropDownList component. When a key is typed, the focus is moved to the respective element, and the value is updated in the component in the open popup state. In the closed popup state, the respective value is updated in the component based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

When the enableVirtualization property is enabled, the skip and take properties provided by the user within the Query class at the initial state or during the actionBegin or actionComplete events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

#### Binding local data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the actionBegin event occurs before the data retrieval starts. Once the data retrieval is successful, the actionComplete event is triggered, indicating that the data fetch process is complete.

In the following example, id column and text column from complex data have been mapped to the value field and text field, respectively.

#### CSHTML

```
@Html.EJS().DropDownList("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).En
ableVirtualization(true).AllowFiltering(false).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "ID",
Text="Text" }).Render()
```

#### VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult virtualscroll()
 {
 ViewBag.data = new Record().RecordModelList();
 return View();
 }
 }
}
```

#### Binding remote data

The DropDownList supports retrieval of data from remote data services with the help of DataManager component. When using remote data, it initially fetches all the data from the server, triggering the actionBegin and actionComplete events, and then stores the data locally. During virtual scrolling,

additional data is retrieved from the locally stored data, triggering the `actionBegin` and `actionComplete` events at that time as well.

The following sample displays the OrderId from the `Orders` Data Service.

#### CSHTML

```
@Html.EJS().DropDownList("records").Placeholder("Select a
item").PopupHeight("200px").DataSource(obj
=>obj.Url("https://ej2services.syncfusion.com/aspnet/development/api/orders"
).CrossDomain(true).Adaptor("WebApiAdaptor")).EnableVirtualization(true).All
owFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "OrderID", Value
= "OrderID" }).Render()
```

#### VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult virtualscroll()
 {
 ViewBag.data = new Record().RecordModelList();
 return View();
 }
 }
}
```

#### Grouping

The DropDownList component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the `groupBy` field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding on virtualization.

The following sample shows the example for Grouping with Virtualization.

#### CSHTML

```
@Html.EJS().DropDownList("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).En
ableVirtualization(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { GroupBy = "Group",
Value = "ID", Text="Text" }).Render()
```

#### VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult virtualscroll()
 {
 ViewBag.data = new Record().RecordModelList();
 return View();
 }
 }
}
```

### Filtering with Virtualization

The DropDownList component supports Filtering with Virtualization. The DropDownList includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the DropDownList sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

### CSHTML

```
@Html.EJS().DropDownList("records").Placeholder("Select a
item").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).En
ableVirtualization(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "ID",
Text="Text" }).Render()
```

### VIRTUALSCROLL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult virtualscroll()
 {
 ViewBag.data = new Record().RecordModelList();
 return View();
 }
 }
}
```

```
}
```

## Grouping

The DropDownList supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

### CSHTML

```
@Html.EJS().DropDownList("Vegetable").Placeholder("Select a
Vegetable").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data)
.Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value =
"Vegetable", GroupBy = "Category" }).Render()
```

### VEGETABLES.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Vegetables
 {
 public string Vegetable { get; set; }
 public string Category { get; set; }
 public string Id { get; set; }
 public List<Vegetables> VegetablesList()
 {
 List<Vegetables> veg = new List<Vegetables>();
 veg.Add(new Vegetables { Vegetable = "Cabbage", Category= "Leafy and
Salad", Id= "item1" });
 veg.Add(new Vegetables { Vegetable = "Chickpea", Category= "Beans",
Id= "item2" });
 veg.Add(new Vegetables { Vegetable = "Garlic", Category= "Bulb and
Stem", Id= "item3" });
 veg.Add(new Vegetables { Vegetable = "Green bean", Category=
"Beans", Id= "item4" });
 veg.Add(new Vegetables { Vegetable = "Horse gram", Category=
"Beans", Id= "item5" });
 veg.Add(new Vegetables { Vegetable = "Nopal", Category= "Bulb and
Stem", Id= "item6" });
 veg.Add(new Vegetables { Vegetable = "Onion", Category= "Bulb and
Stem", Id= "item7" });
 veg.Add(new Vegetables { Vegetable = "Pumpkins", Category= "Leafy
and Salad", Id= "item8" });
 veg.Add(new Vegetables { Vegetable = "Spinach", Category= "Leafy and
Salad", Id= "item9" });
 veg.Add(new Vegetables { Vegetable = "Wheat grass", Category= "Leafy
and Salad", Id= "item10" });
 veg.Add(new Vegetables { Vegetable = "Yarrow", Category = "Leafy and
Salad", Id = "item11" });
 }
 }
}
```

```

 return veg;
 }
}

```

### Customization

The grouping header is also provided with customization option. This allows custom designing using the `groupTemplate` property for both inline and fixed headers as referred here:

[Group Template support to DropDownList.](#)

### Filtering

The DropDownList has built-in support to filter data items when [allowFiltering](#) is enabled. The filter operation starts as soon as you start typing characters in the search box.

To display the filtered items in the popup, filter the required data and return it to the DropDownList via `updateData` method by using the [filtering](#) event.

The following sample illustrates how to query the data source and pass the data to the DropDownList through the `updateData` method in `filtering` event.

### CSHTML

```

@Html.EJS().DropDownList("country").Placeholder("Select a
Country").PopupHeight(
 "230px").DataSource((IEnumerable<object>)ViewBag.data).AllowFiltering(true).
Filtering("onfiltering").Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "Name"
}).Render()
<script>
 function onfiltering(e) {
 var query = new ej.data.Query();
 query = (e.text !== '') ? query.where('Name', 'startswith',
e.text, true) : query;

e.updateData(@Html.Raw(JsonConvert.SerializeObject(ViewBag.data)), query)
 }
</script>

```

### COUNTRIES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Countries
 {
 public string Name { get; set; }
 public string Code { get; set; }
 public List<Countries> CountriesList()
 {
 List<Countries> country = new List<Countries>();

```



```

country.Add(new Countries { Name = "Australia", Code = "AU" });
country.Add(new Countries { Name = "Bermuda", Code = "BM" });
country.Add(new Countries { Name = "Canada", Code = "CA" });
country.Add(new Countries { Name = "Cameroon", Code = "CM" });
country.Add(new Countries { Name = "Denmark", Code = "DK" });
country.Add(new Countries { Name = "France", Code = "FR" });
country.Add(new Countries { Name = "Finland", Code = "FI" });
country.Add(new Countries { Name = "Germany", Code = "DE" });
country.Add(new Countries { Name = "Greenland", Code = "GL" });
country.Add(new Countries { Name = "Hong Kong", Code = "HK" });
country.Add(new Countries { Name = "India", Code = "IN" });
country.Add(new Countries { Name = "Italy", Code = "IT" });
country.Add(new Countries { Name = "Japan", Code = "JP" });
country.Add(new Countries { Name = "Mexico", Code = "MX" });
country.Add(new Countries { Name = "Norway", Code = "NO" });
country.Add(new Countries { Name = "Poland", Code = "PL" });
country.Add(new Countries { Name = "Switzerland", Code = "CH"
});
country.Add(new Countries { Name = "United Kingdom", Code = "GB"
});
country.Add(new Countries { Name = "United States", Code = "US"
});
return country;
}
}
}

```

### Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the DropDownList. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").Filtering("onfiltering").AllowFiltering(true).PopupHeight("200px"
).DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
"ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "ContactName",
Value = "CustomerID" }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
function onfiltering(e) {
 var CBObj = document.getElementById("customers").ej2_instances[0];
 // load overall data when search key empty.
 if (e.text == '' && e.text.length < 3) {
 e.updateData(CBObj.dataSource);
 }
 let query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);

```

```

 query = (e.text !== '' && e.text.length >= 3) ?
 query.where('ContactName', 'startswith', e.text, true) : query;
 e.updateData(CBObj.dataSource, query);
 }
</script>

```

### FILTERLIMIT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult filteringlimit()
 {
 return View();
 }
 }
}

```

### Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a customer").AllowFiltering(true).Filtering("onfiltering").PopupHeight("200px")
.DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
 "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "ContactName",
Value = "CustomerID" }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
 function onfiltering(e) {
 var CBObj = document.getElementById("customers").ej2_instances[0];
 // load overall data when search key empty.
 if (e.text == '')
 e.updateData(CBObj.dataSource);
 else {
 var query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
 query = (e.text !== '') ? query.where('ContactName', 'endswith',
e.text, true) : query;
 e.updateData(CBObj.dataSource, query);
 }
 }

```

```

 }
}
</script>

```

### FILTERTYPE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult filtertype()
 {
 return View();
 }
 }
}

```

### Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the `where` clause.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a customer").Filtering("onfiltering").AllowFiltering(true).PopupHeight("200px").DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "ContactName", Value = "CustomerID" }).Query("new ej.data.Query().from('Customers').select(['ContactName', 'CustomerID']).take(6)").Render()
<script>
 function onfiltering(e) {
 var CBObj = document.getElementById("customers").ej2_instances[0];
 // load overall data when search key empty.
 if (e.text == '')
 e.updateData(CBObj.dataSource);
 else {
 var query = new
ej.data.Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
 query = (e.text != '') ? query.where('ContactName', 'startswith', e.text, false) : query;
 e.updateData(CBObj.dataSource, query);
 }
 }
</script>

```

### CASESENSITIVE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult casesensitive()
 {
 return View();
 }
 }
}
```

### Diacritics Filtering

DropDownList supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

### CSHTML

```
@Html.EJS().DropDownList("list").DataSource((string[])ViewBag.data).Placeholder("e.g: aero").IgnoreAccent(true).AllowFiltering(true).Render()
```

### DIACRITICS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult diacritics()
 {
 ViewBag.data = new string[] { "Aeróbics", "Aeróbics en Agua",
"Aerografía", "Aeromodelaje", "Águilas", "Ajedrez", "Ala Delta", "Álbumes de
Música", "Alusivos", "Análisis de Escritura a Mano" };
 return View();
 }
 }
}
```

### See Also

- [How to limit the search result while filtering](#)
- [How to highlight the matched characters in filtering](#)

- [How to modify the result data using remote data source](#)

### Localization in ASP.NET MVC DropDownList control

The Localization library allows to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the DropDownList.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

### Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the DropDownList and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

### CSHTML

```
@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").Locale("fr-BE").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "ContactName"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(0)").Render()
<script>
 ej.base.L10n.load({
 'fr-BE': {
 'dropdowns': {
 'noRecordsTemplate': "Aucun enregistrement trouvé",
 'actionFailureTemplate': "Modèle d'échec d'action"
 }
 }
 });
</script>
```

### LOCALIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
```

```
public ActionResult localization()
{
 return View();
}
}
```

### See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

### CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}
,
```

#### Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}
,
```

#### Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-
input-focus::after {
```

```
background: #c000ff;
}
```

### Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css

.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {

border-color: #b1bd15;

box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;

}
```

### Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css

.e-input-group.e-control-wrapper .e-input[disabled] {

-webkit-text-fill-color: #0d9133;

}
```

### Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css

.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::after {

background-color: #2319b8;

}

.e-ddl.e-lib.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top {

color: #2319b8;

}
```

### Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css
.e-ddl.e-input-group input.e-input::placeholder {
color: red;
}
`
```

### Customizing the placeholder to add mandatory indicator(\*)

Use the following CSS to add the mandatory indicator \* to the float label element.

```
`css
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after{
content: "*";
color: red;
}
`
```

### Customizing the background color of focus, hover, and active item

Use the following CSS to customize the background color of focus, hover and active item.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
`
```

### Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
`
```



## Accessibility

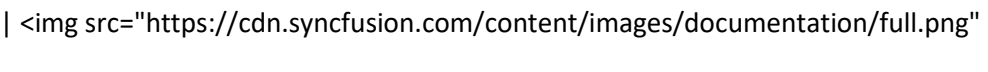
The DropDownList control has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This control is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The DropDownList component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DropDownList component is outlined below.

| Accessibility Criteria | Compatibility |

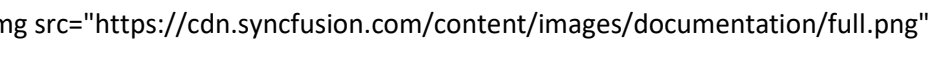
| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div><img alt="Yes" data-bbox="271 822 862 856"/> - All features of the component meet the requirement.</div>

<div><img alt="Intermediate" data-bbox="271 863 791 898"/> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The DropDownList control uses the **Listbox** role, and each list item has an **option** role. The following ARIA attributes denotes the DropDownList state.

#### | Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the DropDownList input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the DropDownList element. |

| aria-disabled | Indicates whether the DropDownList control is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

### Keyboard interaction

You can use the following key shortcuts to access the DropDownList without interruptions.

#### | Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Selects the first item in the DropDownList when no item is selected. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |

| Enter | Selects the focused item, and when it is in an open state the popup list closes. Otherwise, toggles the popup list. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the control. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the control. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Selects the first item. |

| End | Selects the last item. |

**Note:** In the below sample, focus the DropDownList control using alt+t keys.

#### CSHTML

```
@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Re
nder()
<script>
 document.onkeyup = function (e) {
 if (e.altKey && e.keyCode === 84 /* t */) {
 var dropObj = document.getElementById("games"); //to get
dropdown list object
 dropObj.ej2_instances[0].focusIn(); // call the focusIn method
to focus the element.
 }
 };
</script>
```

#### ACCESSIBILITY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult accessibility()
 {
 ViewBag.data = new string[] { "Badminton", "Basketball",
"Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View();
 }
 }
}
```

#### Ensuring accessibility

The DropDownList component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DropDownList component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DropDownList component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## How To

### Add item in between in DropDownList

You can add item in between based on item [index](#). If you add new item without item index, item will be added as last item in list.

#### CSHTML

```
@Html.EJS().Button("first").Content("add item (Hockey) in first").Render()
@Html.EJS().Button("between").Content("add item (Golf) in Between").Render()
@Html.EJS().Button("last").Content("add item (Cricket) in last").Render()
@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Fields(
new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "Game"
}).Render()
<script>
 document.getElementById('first').onclick = () => {
 var dropObject = document.getElementById("games").ej2_instances[0];
 dropObject.addItem({ Game: 'Hockey' }, 0);
 };
 // add item in between
 document.getElementById('between').onclick = () => {
 var dropObject = document.getElementById("games").ej2_instances[0];
 dropObject.addItem({ Game: "Golf" }, 2);
 };
 // add item at last
 document.getElementById('last').onclick = () => {
 var dropObject = document.getElementById("games").ej2_instances[0];
 dropObject.addItem({ Game: "Cricket" });
 };
</script>
```

#### GAMELIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class GameList
 {
 public string Id { get; set; }
 public string Game { get; set; }
 public List<GameList> GameLists()
 {
 List<GameList> game = new List<GameList>();
 game.Add(new GameList { Id = "Game1", Game = "American Football"
 });
 game.Add(new GameList { Id = "Game2", Game = "Badminton" });
 game.Add(new GameList { Id = "Game3", Game = "Basketball" });
 game.Add(new GameList { Id = "Game3", Game = "Basketball" });
 game.Add(new GameList { Id = "Game4", Game = "Cricket" });
 game.Add(new GameList { Id = "Game5", Game = "Football" });
 game.Add(new GameList { Id = "Game6", Game = "Golf" });
 game.Add(new GameList { Id = "Game7", Game = "Hockey" });
 game.Add(new GameList { Id = "Game8", Game = "Rugby" });
 }
 }
}
```

```

 game.Add(new GameList { Id = "Game9", Game = "Snooker" });
 game.Add(new GameList { Id = "Game10", Game = "Tennis" });
 return game;
 }
}

```

### Configure the Cascading DropDownList

The cascading DropDownList is a series of DropDownList, where the value of one DropDownList depends upon another's value. This can be configured by using the [change](#) event of the parent DropDownList. Within that change event handler, data has to be loaded to the child DropDownList based on the selected value of the parent DropDownList.

The following example, shows the cascade behavior of country, state, and city DropDownList. Here, the [dataBind](#) method is used to reflect the property changes immediately to the DropDownList.

### CSHTML

```

<div class="padding-top">
 @Html.EJS().DropDownList("country").Placeholder("Select a
country").PopupHeight(@ViewBag.popupHeight).Change("countrychange").DataSou
rce((IEnumerable<Object>)ViewBag.country).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CountryName",
Value = "CountryId" }).Render()
</div>
<div class="padding-top">
 @Html.EJS().DropDownList("state").Placeholder("Select a
state").Enabled(false).PopupHeight(ViewBag.popupHeight).Change("statechange"
).DataSource((IEnumerable<Object>)ViewBag.state).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "StateName",
Value = "StateId" }).Render()
</div>
<div class="padding-top">
 @Html.EJS().DropDownList("city").Placeholder("Select a
city").Enabled(false).PopupHeight(ViewBag.popupHeight).DataSource((IEnumerableab
le<Object>)ViewBag.cities).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CityName",
Value = "CityId" }).Render()
</div>
<script type="text/javascript">
 function countrychange() {
 // disable the state DropDownList
 var countryObj =
document.getElementById('country').ej2_instances[0];
 var state = document.getElementById('state').ej2_instances[0];
 var city = document.getElementById('city').ej2_instances[0];
 state.enabled = true;
 //frame the query based on selected value in country DropDownList.
 var tempQuery = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
 // set the framed query based on selected value in country
DropDownList.
 state.query = tempQuery;
 // set null value to state DropDownList text property
 state.text = null;
 // bind the property changes to state DropDownList
 }

```

```

 state.dataBind();
 // set null value to city DropDownList text property
 city.text = null;
 // disable the city DropDownList
 city.enabled = false;
 // bind the property changes to City DropDownList
 city.dataBind();
 }
 function statechange() {
 var stateObj = document.getElementById('state').ej2_instances[0];
 var city = document.getElementById('city').ej2_instances[0];
 city.enabled = true;
 //Query the data source based on state DropDownList selected value
 var tempQuery = new ej.data.Query().where('StateId', 'equal',
stateObj.value);
 //set the framed query based on selected value in city DropDownList.
 city.query = tempQuery;
 //clear the existing selection
 city.text = null;
 //bind the property change to city DropDownList
 city.dataBind();
 }
</script>

```

## COUNTRY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Country
 {
 public string CountryName { get; set; }
 public string CountryId { get; set; }
 public List<Country> CountryList()
 {
 List<Country> country = new List<Country>();
 country.Add(new Country() { CountryName = "Australia", CountryId
= "2" });
 country.Add(new Country() { CountryName = "United States",
CountryId = "1" });
 return country;
 }
 }
}

```

### Preselect value from model in Cascading DropDownList

The cascading DropDownList is a series of DropDownList, where the value of one DropDownList depends upon another's value. Values can be preselected in these DropDownList from model. Use the dataManager to perform the filtering operation on the JSON data for cascading through create event.

## CSHTML

```

<div class="padding-top">
 @Html.EJS().DropDownListFor(model =>
model.CountryId).Placeholder("Select a
country").Width("300").PopupHeight("auto").Change("countrychange").DataSourc
e((IEnumerable<Object>)ViewBag.country).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CountryName",
Value = "CountryId" }).Render()
</div>
<div class="padding-top">
 @Html.EJS().DropDownListFor(model => model.StateId).Placeholder("Select
a
state").Width("300").PopupHeight("auto").Created("stateCreated").DataSource(
(IEnumerable<object>)ViewBag.state).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "StateName",
Value = "StateId" }).Render()
</div>
<script type="text/javascript">
 var totalData;
 function stateCreated() {
 // disable the state DropDownList
 var countryObj =
document.getElementById('CountryId').ej2_instances[0];
 var state = document.getElementById('StateId').ej2_instances[0];
 totalData = state.dataSource
 //frame the query based on selected value in country DropDownList.
 var dataManager = new ej.data.DataManager(totalData);
 var tempQuery = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
 // set the framed query based on selected value in country
DropDownList.
 state.dataSource = tempQuery.executeLocal(dataManager);
 }
 function countrychange() {
 // disable the state DropDownList
 var countryObj =
document.getElementById('CountryId').ej2_instances[0];
 var state = document.getElementById('StateId').ej2_instances[0];
 state.enabled = true;
 var dataManager = new ej.data.DataManager(totalData);
 //frame the query based on selected value in country DropDownList.
 var tempQuery = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
 // set the framed query based on selected value in country
DropDownList.
 state.dataSource = tempQuery.executeLocal(dataManager);
 state.dataBind();
 }
</script>

```

## COUNTRY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models

```

```

{
 public class Country
 {
 public string CountryName { get; set; }
 public string CountryId { get; set; }
 public List<Country> CountryList()
 {
 List<Country> country = new List<Country>();
 country.Add(new Country() { CountryName = "Australia", CountryId
= "2" });
 country.Add(new Country() { CountryName = "United States",
CountryId = "1" });
 return country;
 }
 }
}

```

### Model binding with nested class

The following example demonstrate on how to model bind data to DropDownList with nested classes in controller.

#### CSHTML

```

<div class="container">
 @Html.EJS().DropDownList(Html.IdFor(m =>
m.SelectedAccountNumber).ToString()).DataSource(Model.BeneficiaryList).Width
("300").Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings()
 {
 Text = "PersonName.FullName",
 Value = "AccountNumber"
 }).ItemTemplate("${PersonName.FullName} -
${AccountNumber}").ValueTemplate("${PersonName.FullName} -
${AccountNumber}").Render()
</div>

```

#### NESTEDDATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult nesteddata()
 {
 BeneficiaryViewModel model = new BeneficiaryViewModel();
 List<Person> person = new List<Person>();
 person.Add(new Person { AccountNumber = "5000102394555",
PersonName = new Name { FirstName = "Mary", LastName = "Elizabeth", FullName
= "Mary Elizabeth" } });

```



```

 person.Add(new Person { AccountNumber = "5000102394556",
 PersonName = new Name { FirstName = "Liz ", LastName = "Smith", FullName =
 "Liz Smith" } });
 person.Add(new Person { AccountNumber = "5000102394557",
 PersonName = new Name { FirstName = "Ada", LastName = "Maria", FullName =
 "Ada Maria" } });
 person.Add(new Person { AccountNumber = "5000102394558",
 PersonName = new Name { FirstName = "Elizabeth", LastName = "Smith",
 FullName = "Elizabeth Smith" } });
 person.Add(new Person { AccountNumber = "5000102394559",
 PersonName = new Name { FirstName = "George", LastName = "David", FullName =
 "George David" } });
 person.Add(new Person { AccountNumber = "5000102394560",
 PersonName = new Name { FirstName = "Nancy", LastName = "Rose", FullName =
 "Nancy Rose" } });
 model.BeneficiaryList = person;
 model.SelectedAccountNumber = "persondetails";
 return View(model);
 }
}

public class BeneficiaryViewModel
{
 public string SelectedAccountNumber { get; set; }
 public List<Person> BeneficiaryList { get; set; }
}

public class Person
{
 public string AccountNumber { get; set; }
 public Name PersonName { get; set; }
}

public class Name
{
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string FullName { get; set; }
}
}

```

### Clear the selected item in DropDownList

You can clear the selected item in the below two different ways.

By clicking on the **clear icon** which is shown in DropDownList element, you can clear the selected item in DropDownList through **interaction**. By using [showClearButton](#) property, you can enable the clear icon in DropDownList element.

Through **programmatic** you can set **null** value to anyone of the index, text or value property to clear the selected item in DropDownList.

### CSHTML

```

@Html.EJS().Button("btn").Content("Set Null value to property").Render()
@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Re
nder()
<script>
 document.getElementById('btn').onclick = () => {

```

```

 var dropObj = document.getElementById("games"); //to get dropdown
 list object
 dropObj.value = null;
 };
</script>

```

### CLEARSELECTION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult tooltip()
 {
 ViewBag.data = new string[] { "American Football", "Badminton",
 "Basketball", "Cricket", "Football", "Golf", "Hockey", "Rugby", "Snooker",
 "Tennis" };
 return View();
 }
 }
}

```

### Close the popup on scroll

By using the `hidePopup` method in DropDownList, you can close the popup on scroll when triggered the windows scroll event.

### CSHTML

```

@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Re
nder()
<script>
 window.onscroll = () => {
 var dropObj = document.getElementById("games"); //to get dropdown
 list object
 dropObj.ej2_instances[0].hidePopup(); // hide the popup using
 hidePopup method
 }
</script>

```

### CLOSEPOPUP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers

```

```

{
 public class DropDownListController : Controller
 {
 public IActionResult closepopup()
 {
 ViewBag.data = new string[] { "American Football", "Badminton",
 "Basketball", "Cricket", "Football", "Golf", "Hockey", "Rugby", "Snooker",
 "Tennis" };
 return View();
 }
 }
}

```

### Disable the Fixed group header in DropDownList

The following example demonstrate about how to disable the Fixed group header in DropDownList through CSS by using `visibility` attribute.

#### CSHTML

```

@Html.EJS().DropDownList("Vegetable").Placeholder("Select a
Vegetable").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value =
"Vegetable", GroupBy = "Category" }).Render()
<style>
 .e-ddl .e-dropdownbase .e-fixed-head {
 visibility: hidden;
 }
</style>

```

#### VEGETABLES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Vegetables
 {
 public string Vegetable { get; set; }
 public string Category { get; set; }
 public string Id { get; set; }
 public List<Vegetables> VegetablesList()
 {
 List<Vegetables> veg = new List<Vegetables>();
 veg.Add(new Vegetables { Vegetable = "Cabbage", Category= "Leafy and
 Salad", Id= "item1" });
 veg.Add(new Vegetables { Vegetable = "Chickpea", Category= "Beans",
 Id= "item2" });
 veg.Add(new Vegetables { Vegetable = "Garlic", Category= "Bulb and
 Stem", Id= "item3" });
 veg.Add(new Vegetables { Vegetable = "Green bean", Category=
 "Beans", Id= "item4" });
 veg.Add(new Vegetables { Vegetable = "Horse gram", Category=
 "Beans", Id= "item5" });
 }
 }
}

```

```

 veg.Add(new Vegetables { Vegetable = "Nopal", Category= "Bulb and Stem", Id= "item6" });
 veg.Add(new Vegetables { Vegetable = "Onion", Category= "Bulb and Stem", Id= "item7" });
 veg.Add(new Vegetables { Vegetable = "Pumpkins", Category= "Leafy and Salad", Id= "item8" });
 veg.Add(new Vegetables { Vegetable = "Spinach", Category= "Leafy and Salad", Id= "item9" });
 veg.Add(new Vegetables { Vegetable = "Wheat grass", Category= "Leafy and Salad", Id= "item10" });
 veg.Add(new Vegetables { Vegetable = "Yarrow", Category = "Leafy and Salad", Id = "item11" });
 return veg;
 }
}

```

### Highlight the matched character in filtering

By using the **highlightSearch** method, you can highlight the matched character in DropDownList filtering.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a customer").Filtering("filtering").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true).AllowFiltering(true).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "CustomerID", Text = "ContactName"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName', 'CustomerID']).take(6)").Render()
<script>
 var queryString;
 document.addEventListener('DOMContentLoaded', function () {
 var ddlObj = document.getElementById("customers")
 ddlObj.ej2_instances[0].fields = {
 text: "ContactName", value: "CustomerID", itemCreated: function
(e) {
 new ej.dropdowns.highlightSearch(e.item, queryString, true, 'StartsWith');
 }
 }, false);
 function filtering(e) {
 var data = document.getElementById('customers').ej2_instances[0];
 // take text for highlight the character in list items.
 queryString = e.text;
 var query = new
ej.data.Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
 //frame the query based on search string with filter type.
 query = (e.text !== '') ? query.where('ContactName', 'startswith', e.text, true) : query;

```

```
//pass the filter data source, filter query to updateData method.
e.updateData(data.dataSource, query);
}
</script>
```

### HIGHLIGHT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult highlight()
 {
 return View();
 }
 }
}
```

### Show the list items with icons

You can render **icons** to the list items by mapping the [iconCss](#) field. This **iconCss** field creates a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with **iconCss** field.

### CSHTML

```
@Html.EJS().DropDownList("icons").Placeholder("Select a social
media").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.icondat
a).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value =
"SocialMediaName", IconCss = "Class" }).Render()
<style>
 @@font-face {
 font-family: 'Socialicons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAAwAgTlMvMvltCfsAAAEoAAAVmNtYXCnKKeOAAABrAAAAEhnbHlml19
XagAAAgwAABhQaGvHZA8dCeEAAADQAAAAANmhoZWEIUQQMAAAARAAAACRobXR4LAAAAAAAYAAAA
sbG9jYR3AIwwAAAH0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGX5D00
AABY0AAAAkwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAACwABAAAAAQAA+iTiP18
PPPUACwQAAAAANYFYngAAAAAlgVieAAAAAAD9AP0AAAACAACAAAAAAAAAAAAAALafQACwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCQQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaawAAABQAAwABAAAAFAA
EADQAAAAEAAQAAQAApwn//wAApWd//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAAiQ
CzgOMBU4F/gZYB9QIcAo+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fHQEfDhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLwYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kIw8FAYkFEgQDAyQDAQECAYIBAQMSEgkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgWKCQQEBQICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwCFBQUEAwICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRUVFBU
VDw4ODg0NDAwLCwoKCQkICAcGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwWNDQ0ODg4PDw8QEBA
```

QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwwNDQ4ODxAREhISEhMTEExMUEXQUFRQ  
VGXsaGgcIBwfxNgEBAQ8KCgoIBwcGBQUDAwIBAQECAwMDBQUBGcHCAgICgkKCwsMDAwNDg0ODw8  
PEBAQEhISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCAcH/aAQEB0cGhgWFBIRDgw  
LCaCEAwICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQEA8ODhAQDQ0LCQg  
HBgQCAgICBAQEBgYGCAGJCQoLCwwMDQ4ODg8PEBAREBESEhISExITGBcZGRgZGBgXAU4CagMEXAk  
FBAQFBCQCAwMGHcKFQkMIwIBAwOYAwIBKQECBSgFBgULCgkHBgMBAQEDAwcICgMDg8PEhITFBu  
WGBgSEyYJCAgIBwcHDBEGAgEBAagEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhE  
SEREREBEPEA8QDhMEBASFNgEBAQEJCACGBQMCAQEDAwUGCAGHCAgJCQoKCwsMDA0NDQ4ODg8ODxA  
PEA8QEBAQEBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICACGBgUFAwMDAQEBAQE6RISExISExISEhM  
SEhESERERERAE8QDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVncFBAQVEBEREhESEhI  
TExMTFBMUEhEREBEQEBApDw8PDg4ODQ0MDAwLCwoKCgkICACHBWUGBQQDAgIBAQEBAgIDBAQFBQc  
GCAgICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUfHcYGRschr8fIiIjFBMTEhMSEhIREHEREBEQEAM  
EAWt1YQINCAyEAgIEBAUGBgICQoKDAwNDg8PERUVFhCWGBcYGBkZGRkaGxOTehISEHEREBAQDw8  
ODg4MDQwLCgoKCQgHBwYFBQMDAwEBAgMFBQgIAAAAAAEAAAAAzoD9ACWAAATDwYVERUfHTsBPw4  
9AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwwMDA0NDQ0ODg4PDw8  
QEBDQCgsKCQkJCAgHBWUEBAICAIEBAUHBwgICQkJCgsK0QoKCgoJCAgIBwcFBAMDAQEBKQoJCgg  
JCAgHBWUFBAQCAQEBAQIEBAUFBgHCAkJCQkK/tcCagMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgs  
KDAwMDf4LExMTEhESEREQEBApDw4PDQ4MDAOKCQgIBgYEBAMCAgEBAwQFBwcJCQoKCwsMDA0NDAs  
MCwoKCQkHBWUEAwEBAQEDBAUGCAGJCgoLCwwMDVgCAwMFBGcICQkJCgsLCwwLDAsKCgoJCAgHBgU  
EAgIBtQ0MDAsLCgoJCQcHBQQDAQEBAQMEBQYIAAABAAAAAP0A90AqAAAAT8DMx8MHQEPDSsBLxo  
PCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBWUFBQQDAwICAgEDBQoPEXyWFAcLCgQFBAUFBAU  
FBQUJCQkJCyQEBAQUGBwcICAKKCgsLDQwODQ8RERQfI5IvNRMGBWYGBWYGBgYGDASMWAcICAgICQk  
JCQkKCgoLCgsJEXITFBQVFRYWFIMkJTEWFRQREQ8NDAsJBWYFAwEBAQEDBAUHBwkJCwwNDhAQEGs  
YGBYWFBSQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBQUHBGcICBMSDxEIiUoIXsOCgcCAQIEBAYICBU  
bHyU37xQTERAPDQwKCQgGBQMCAQEDBgLDhofkUInCwIBAgQEBwcJCwscISf+pBYUEXIQDg4LCwk  
IBgUDAgEDBACJDA0REhQXJysxRiEhIB4eHRwbGhkZFxcVFRoXfHqTERAODQwKCACGBAMBAQMFBwk  
MDQ8REXUXGhsdAAUAAAAA/ED9ABCAKoA6wESAYQAAEdAQ8NKwEvDjU/EB8OJR0Bhw8hPw8TLwM  
hHwUVDxEvEzU/CSchDwMFFR8PPw8vDw8OAR8HFQ8JIY8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM  
9Ai8BIw8EFQ8DIY8CNw8KFxUfASUzPwgZHWkhPwI1LxAlDwICkAMDBQcHCQkKDAwMDQ4ODwwMCws  
LCgoJDwsJCAYFAgECAwQGBGcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBGkJDAwODwg  
LERITewJpExMSEhEQDw4MDAUJBWYEAgEBAgEF/uYOCwkGBAICBAYHCQsMDg8QERETEXQTFRQVFBQ  
UFBMTEhAPDg0LCQgGBAMCAgEBAwMEBQYHCAKc/uoFAGEBASwBAWUGCQoLDQ0PERESEhQUFBQTEhE  
QEA4MDAKJBgUDAQEDBQcICgsNDg8QERISFBQUFBMSERAQDgWMCggHBAMCPAYGBgQEAWEBABQEBAM  
EBAQFBmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFeAKdAwECBAQEBAQDwI  
CAWUWAwIBAQQQDwwLCQgFAwEBAQEEATEEBBYUFYRYWfxcYFxcXfXyVFBgFBQYBjWYCAgICBAYHCQo  
LDA4ODxAIERIR/d8FAQIB9wchDg0NDAwLCgkIBwYFBAICAQEBQYGDQwODg8REBINDQwMCwsKCgk  
IBwcGBAQCAQEBAgQFBggICgoLDQ0NDg9929sUEXISERAPDgwMBQkHBgQCAQMFBGkJDAwODwgQERI  
TEWHBGMBARYXFxcXfxcWFhUUFBMREQ8ODAsJCAYEAwIBAgQFBwkKDA0PDxEREXQPEA8PDw8PDw8  
ODw4ODg4OAQEBAQKPCgoUEhIREBANDQsKCAcFAwEBAwUHCAoLDQ4PEBESExQUFBMTEhEQDw4NCwo  
IBWUDAQEDBQcICgsNDg8QEHITewGSAQICBAUFBgdsBQQFBAQDAgIBAQECAgQFBQYHawCHBgUDAgE  
BR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDAWMG0ggEAQICAgTKAQ00EBASEhQ  
VEiRdAgIBAQITDg0JCAYDAQFBwOMDhQCAQEBAQNujBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAM  
AAAAA/QD3QADAFcAlwAAZMriWUVIzclIXEzET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQH  
W1gIDBQgKCwCHBgJCQoKCw4NDAsKCAgHBWUEBAICAQHWAQICAgQDBQUBGcYHBwcJCAkJCgoKCws  
LDBgZGhQUEREPDg0MCwoJCQ79xAEBawMFBgYHCAkKCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAY  
GCAgJCgoLDQwODQ0MDAOKCQkHBWYEBAMBIGKFWwICW/17AXcUDA0ODgwGBQUEBAMCAQEBAgMFBQc  
ICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQwMCwoKCQkICACGBgUFBAMGAwEBAgMEBgyHCAgICQkSARI  
MCwsKCgkICAGGBQEAWEBABQEDBAUFBgICAKKCgsLDAsLCwsJCggIBWYGBAQDAQEBAQYHBAWYBgWg  
ICGkKCgsAAAAAABAAAAA/PuA/QARgAAEXEVHwYhESM1MzU/DzMWIw8GFTMVIxehPwYRLwYhDwYSAGQ  
FBWgKCGHPb24BAwMGBggJCgsMDQ0ODwgPlUcLCwkIBgQDe3sBBQoKCAcFBAICBAUHCAOK/IUKCgk  
HBWQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBWUEAgGFAwQHCAkKDDOF/mUDBAYICQoLA4I  
LCgkIBgQDAQQFBwgKCwAAAAAGAAAAAP0A/QAOABEAiABBQEQAuWAAAEPCR0Bhw07AT8NPQEVC  
PASUVMxUjFSM1IzUzNSUPBRUfDTsBPww1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIH1Y8NPQE  
/DwEVHw8hPw8RITChLw8hDw4BCgMTCwsFBAQEAgICAwQGBGcICgoLDAwODg8NDQwLCgkICAYGBQQ  
DAwEBAQIDBAgMDiYRNw0B9nR0TXNz/kAFawMDAQIBAgMDBAQGBGcICQkKDAsIBwcHBWYFBQYFAwM  
BAQECAwMEBQYGBwgJCQoLDAcIBwcHBwX+MTAQDggIAwICAQEBAQEDAwMICgsMDAsGAgEBAQECAwY  
iGQoFCQcDAgIBAwQFCAGLDA0PERITFRYYFRISEA8NDAsKCAcGBAMCAQEBAwUHCQsOERMUFB0xCAC

Copyright © 2001 -2024 Syncfusion Inc.

```

AAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASBAgEDAQQBBQEGAQcBCAEJAQoBCwE
MAAh3aGF0c2FwcAd0d2l0dGVyBXZpbWVvCWluc3RhZ3JhbQhsaW5rZWRpbGhmYWNlYm9vawtnb29
nbGUtcGxlcwZ0dWlibHIIc2t5cGUtMDEIeW9ldHVizTEAAAA=) format('truetype');
 font-weight: normal;
 font-style: normal;
}
.e-list-icon {
 font-family: 'Socialicons' !important;
 color: rgba(0, 0, 0, .57);
}
.twitter:before {
 content: "\a701";
}
.vimeo:before {
 content: "\a702";
}
.youtube:before {
 content: "\a709";
}
.whatsapp:before {
 content: "\a700";
}
.skype:before {
 content: "\a708";
}
.instagram:before {
 content: "\a703";
}
.google-plus:before {
 content: "\a706";
}
.facebook:before {
 content: "\a705";
}
.tumblr:before {
 content: "\a707";
}
.linkedin:before {
 content: "\a704";
}
}
</style>

```

### SOCIALMEDIA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class SocialMedia
 {
 public string Class { get; set; }
 public string SocialMediaName { get; set; }
 public string Id { get; set; }
 public List<SocialMedia> SocialMediaList()
 }
}

```



```

 {
 List<SocialMedia> media = new List<SocialMedia>();
 media.Add(new SocialMedia{ Class= "facebook", SocialMediaName=
"Facebook", Id= "media1" });
 media.Add(new SocialMedia{ Class= "google-plus",
SocialMediaName= "Google Plus", Id= "media2" });
 media.Add(new SocialMedia{ Class= "instagram", SocialMediaName=
"Instagram", Id= "media3" });
 media.Add(new SocialMedia{ Class= "linkedin", SocialMediaName=
"LinkedIn", Id= "media4" });
 media.Add(new SocialMedia{ Class= "skype", SocialMediaName=
"Skype", Id= "media5" });
 media.Add(new SocialMedia{ Class= "tumblr", SocialMediaName=
"Tumblr", Id= "media6" });
 media.Add(new SocialMedia{ Class= "twitter", SocialMediaName=
"Twitter", Id= "media7" });
 media.Add(new SocialMedia{ Class= "vimeo", SocialMediaName=
"Vimeo", Id= "media8" });
 media.Add(new SocialMedia{ Class= "whatsapp", SocialMediaName=
"WhatsApp", Id= "media9" });
 media.Add(new SocialMedia { Class = "youtube", SocialMediaName=
"YouTube", Id = "media10" });
 return media;
 }
}

```

### Do incremental search in the DropDownList

DropDownList supports incremental search, by default. You can search the list item by focusing the DropDownList and typing the characters in it. The closely matched items are selected sequentially.

**Note:** If the same key is searched once again, the next matched item is selected.

### Modify the result data before passing to DropDownList when binding remote data source

When binding the remote data source, by using the [actionComplete](#) event, you can modify the result data before passing it to DropDownList.

### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").ActionComplete("actionComplete").PopupHeight("200px").DataSource(
dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Text = "ContactName",
 Value = "CustomerID"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
function actionComplete(e) {
 // initially result contains 6 items
 console.log("Before modified the result: " + e.result.length);

```

```

 // remove first 2 items from result.
 e.result.splice(0, 2);
 // now displays the result count is 4.
 console.log("After modified the result: " + e.result.length);
 }
</script>

```

### **MODIFYDATA.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult modifydata()
 {
 return View();
 }
 }
}

```

Preselect the list items in multiple cascading DropDownList

### **CSHTML**

```

<div class="padding-top">
 @Html.EJS().DropDownList("country").Placeholder("Select a
country").PopupHeight(@ViewBag.popupHeight).Change("countrychange").DataSou
rce((IEnumerable<Object>)ViewBag.country).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CountryName",
Value = "CountryId" }).Render()
</div>
<div class="padding-top">
 @Html.EJS().DropDownList("state").Placeholder("Select a
state").Enabled(false).PopupHeight(ViewBag.popupHeight).Change("statechange"
).DataSource((IEnumerable<Object>)ViewBag.state).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "StateName",
Value = "StateId" }).Render()
</div>
<div class="padding-top">
 @Html.EJS().DropDownList("city").Placeholder("Select a
city").Enabled(false).PopupHeight(ViewBag.popupHeight).DataSource((IEnumerableab
le<Object>)ViewBag.cities).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CityName",
Value = "CityId" }).Render()
</div>
<script type="text/javascript">
 function countrychange() {
 // disable the state DropDownList
 var countryObj =
document.getElementById('country').ej2_instances[0];

```

```

var state = document.getElementById('state').ej2_instances[0];
var city = document.getElementById('city').ej2_instances[0];
state.enabled = true;
//frame the query based on selected value in country DropDownList.
var tempQuery = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
// set the framed query based on selected value in country
DropDownList.
state.query = tempQuery;
// set null value to state DropDownList text property
state.text = null;
// bind the property changes to state DropDownList
state.dataBind();
// set null value to city DropDownList text property
city.text = null;
// disable the city DropDownList
city.enabled = false;
// bind the property changes to City DropDownList
city.dataBind();
}
function statechange() {
var stateObj = document.getElementById('state').ej2_instances[0];
var city = document.getElementById('city').ej2_instances[0];
city.enabled = true;
//Query the data source based on state DropDownList selected value
var tempQuery = new ej.data.Query().where('StateId', 'equal',
stateObj.value);
//set the framed query based on selected value in city DropDownList.
city.query = tempQuery;
//clear the existing selection
city.text = null;
//bind the property change to city DropDownList
city.dataBind();
}
</script>

```

## COUNTRY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Country
 {
 public string CountryName { get; set; }
 public string CountryId { get; set; }
 public List<Country> CountryList()
 {
 List<Country> country = new List<Country>();
 country.Add(new Country() { CountryName = "Australia", CountryId
= "2" });
 country.Add(new Country() { CountryName = "United States",
CountryId = "1" });
 return country;
 }
 }
}

```

```

 }
}
}

```

Get the total count of data when remote data bind with DropDownList

Before control rendering, you can get the total items count by using [actionComplete](#) event with its result arguments. After rendering this control, you can get the total items count by using [getItems](#) method.

### CSHTML

```

@Html.EJS().Button("btn").Content("Get Items").Render()
@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("http://services.odata.org/V4/Northwind/Northwind.svc/").Adap
tor("ODataV4Adaptor").CrossDomain(true)).Fields(new
DropDownListFieldSettings
{
 Text = "ContactName",
 Value = "CustomerID"
}).Query((string)ViewBag.query).Render()
<script>
function actionComplete (e) {
 // get total items count
 console.log("Total items count: " + e.result.length);
 var element = document.createElement('p');
 element.innerText = "Total items count: " + e.result.length;
 document.getElementById('event').append(element);
}
document.getElementById('btn').onclick = () => {
 var Obj = document.getElementById("customers").ej2_instances[0];
 // get items count using getItems method
 console.log("Total items count: " + Obj.getItems().length);
 let element = document.createElement('p');
 element.innerText = "Total items count: " + Obj.getItems().length;
 document.getElementById('event').append(element);
};
</script>

```

### TOTALCOUNT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class DropDownListController : Controller
 {
 public ActionResult totalcount()
 {
 return View();
 }
 }
}

```

```
}
}
```

## Remove an item from DropDownList

### CSHTML

```
@Html.EJS().Button("first").Content("Remove 0th Item").Render()
@Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Fi
elds(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "Game"
}).Render()
<script>
 document.getElementById('first').onclick = () => {
 // create DropDownList object
 let obj = document.getElementById('games');
 if (obj.ej2_instances[0].list) {
 // Remove the selected value if 0th index selected
 var dropObject = document.getElementById("games");
 var dropDownListObject = dropObject.ej2_instances[0];
 if (dropDownListObject.index === 0) {
 dropDownListObject.value = null;
 dropDownListObject.dataBind();
 }
 // remove first item in list
 (obj.ej2_instances[0].list.querySelectorAll('li')[0]).remove();
 if (!obj.ej2_instances[0].list.querySelectorAll('li')[0]) {
 dropDownListObject.dataSource = [];
 // enable the nodata template when no data source is empty.
 obj.ej2_instances[0].list.classList.add('e-nodata');
 }
 }
 else {
 // remove first item in list
 dropDownListObject.dataSource.splice(0, 1);
 }
 };
</script>
```

### GAMELIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class GameList
 {
 public string Id { get; set; }
 public string Game { get; set; }
 public List<GameList> GameLists()
 {
 List<GameList> game = new List<GameList>();
 game.Add(new GameList { Id = "Game1", Game = "American Football"
 });
 }
}
```

```

 game.Add(new GameList { Id = "Game2", Game = "Badminton" });
 game.Add(new GameList { Id = "Game3", Game = "Basketball" });
 game.Add(new GameList { Id = "Game3", Game = "Basketball" });
 game.Add(new GameList { Id = "Game4", Game = "Cricket" });
 game.Add(new GameList { Id = "Game5", Game = "Football" });
 game.Add(new GameList { Id = "Game6", Game = "Golf" });
 game.Add(new GameList { Id = "Game7", Game = "Hockey" });
 game.Add(new GameList { Id = "Game8", Game = "Rugby" });
 game.Add(new GameList { Id = "Game9", Game = "Snooker" });
 game.Add(new GameList { Id = "Game10", Game = "Tennis" });
 return game;
 }
}

```

### Limit the search result on filtering

#### CSHTML

```

@Html.EJS().DropDownList("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true).AllowFiltering(true).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings
{
 Value = "ContactName",
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
 function onfiltering(e) {
 var CBObj =
document.getElementById("customers").ej2_instances[0];
 // load overall data when search key empty.
 if (e.text == '' && e.text.length < 3) {
 e.updateData(CBObj.dataSource);
 }
 let query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
 query = (e.text != '' && e.text.length >= 3) ?
query.where('ContactName', 'startswith', e.text, true) : query;
 e.updateData(CBObj.dataSource, query);
 }
</script>

```

#### LIMITSEARCH.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers

```

```
{
 public class DropDownListController : Controller
 {
 public ActionResult limitsearch()
 {
 return View();
 }
 }
}
```

### DropDownList options with tooltip

You can achieve this behavior by using **ej2-tooltip** component. When the mouse hovers on the DropDownList option, the tooltip displays some details related to hovered list item.

### CSHTML

```
@Html.EJS().DropDownList("country").Placeholder("Select a
country").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data)
.Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text =
"Name", Value = "Code" }).Render()
<script>
 var tooltip = new ej.popups.Tooltip({
 // default content of tooltip
 content: 'Loading...',
 // set target element to tooltip
 target: '.e-list-item',
 // set position of tooltip
 position: 'top center',
 // bind beforeRender event
 beforeRender: onBeforeRender
 });
 tooltip.appendTo('body');
 function onBeforeRender(args) {
 // get the target element
 var listElement = document.getElementById('country');
 var result = listElement.ej2_instances[0].dataSource;
 var i;
 for (i = 0; i < result.length; i++) {
 if (result[i].Name === args.target.textContent) {
 this.content = result[i].Name;
 this.dataBind();
 break;
 }
 }
 }
</script>
```

### COUNTRIES.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
```

```

public class Countries
{
 public string Name { get; set; }
 public string Code { get; set; }
 public List<Countries> CountriesList()
 {
 List<Countries> country = new List<Countries>();
 country.Add(new Countries { Name = "Australia", Code = "AU" });
 country.Add(new Countries { Name = "Bermuda", Code = "BM" });
 country.Add(new Countries { Name = "Canada", Code = "CA" });
 country.Add(new Countries { Name = "Cameroon", Code = "CM" });
 country.Add(new Countries { Name = "Denmark", Code = "DK" });
 country.Add(new Countries { Name = "France", Code = "FR" });
 country.Add(new Countries { Name = "Finland", Code = "FI" });
 country.Add(new Countries { Name = "Germany", Code = "DE" });
 country.Add(new Countries { Name = "Greenland", Code = "GL" });
 country.Add(new Countries { Name = "Hong Kong", Code = "HK" });
 country.Add(new Countries { Name = "India", Code = "IN" });
 country.Add(new Countries { Name = "Italy", Code = "IT" });
 country.Add(new Countries { Name = "Japan", Code = "JP" });
 country.Add(new Countries { Name = "Mexico", Code = "MX" });
 country.Add(new Countries { Name = "Norway", Code = "NO" });
 country.Add(new Countries { Name = "Poland", Code = "PL" });
 country.Add(new Countries { Name = "Switzerland", Code = "CH"
 });
 country.Add(new Countries { Name = "United Kingdom", Code = "GB"
 });
 country.Add(new Countries { Name = "United States", Code = "US"
 });
 return country;
 }
}

```

Detect whether the value change happened by manual or programmatic

You can check whether the value change happened by manual or programmatic by using [change](#) event argument, that argument name is `isInteracted`.

### CSHTML

```

@Html.EJS().Button("change").Content("Change the value dynamically").Render()
<div class="control-wrapper">
 <div id="default" style='padding-top:75px;'>
 @Html.EJS().DropDownList("employee").Placeholder("Select a employee").Index(2).Change("onChange").PopupHeight("200px").DataSource((IEnumerableable<object>)ViewBag.data).Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "Name" })
 .Render()
 </div>
</div>
<p id='event'> </p>
<script>
 document.getElementById('change').onclick = () => {
 var dropObject =
 document.getElementById("employee").ej2_instances[0];

```



```

 dropObject.value = 'Andrew Fuller';
 };
 function onChange(args) {
 let element = document.createElement('p');
 document.getElementById("event").innerHTML = "";
 if (args.isInteracted) {
 element.innerText = 'Changes happened by Interaction';
 }
 else {
 element.innerText = 'Changes happened by programmatic';
 }
 document.getElementById('event').append(element);
 }
</script>

```

## EMPLOYEES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class Employees
 {
 public string Name { get; set; }
 public string Eimg { get; set; }
 public string Designation { get; set; }
 public string Country { get; set; }
 public List<Employees> EmployeesList()
 {
 List<Employees> emp = new List<Models.Employees>();
 emp.Add(new Employees { Name= "Andrew Fuller", Eimg= "7",
Designation = "Team Lead", Country= "England"});
 emp.Add(new Employees { Name= "Anne Dodsworth", Eimg= "1",
Designation = "Developer", Country= "USA"});
 emp.Add(new Employees { Name= "Janet Leverling", Eimg= "3",
Designation = "HR", Country= "USA"});
 emp.Add(new Employees { Name= "Laura Callahan", Eimg= "2",
Designation = "Product Manager", Country= "USA"});
 emp.Add(new Employees { Name= "Margaret Peacock", Eimg= "6",
Designation = "Developer", Country= "USA"});
 emp.Add(new Employees { Name= "Michael Suyama", Eimg= "9",
Designation = "Team Lead", Country= "USA"});
 emp.Add(new Employees { Name= "Nancy Davolio", Eimg= "4",
Designation = "Product Manager", Country= "USA"});
 emp.Add(new Employees { Name= "Robert King", Eimg= "8",
Designation = "Developer ", Country= "England"});
 emp.Add(new Employees { Name= "Steven Buchanan", Eimg= "10",
Designation = "CEO", Country= "England" });
 return emp;
 }
 }
}

```

Whether each list items hold unique value

Yes, the value for each list items should be unique.

### DropDownListFor

The DropDownListFor control can be rendered by passing values and data from the model. The selected values can be retrieved during form submit using the post method.

#### CSHTML

```
@using (Html.BeginForm())
{
 @Html.EJS().DropDownListFor(model => model.Value).Placeholder("Select a
Country").DataSource((IEnumerable<object>)ViewBag.data).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "Name"
}).Render()
 <div id="submitButton">
 @Html.EJS().Button("btn").Content("Post").Render()
 </div>
}
```

#### FOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class AutoCompleteController : Controller
 {
 Countries model = new Countries();
 public ActionResult Index()
 {
 ViewBag.data = new Countries().CountriesList();
 model.Value = "Cameroon";
 return View(model);
 }
 [HttpPost]
 public ActionResult Index(Countries model)
 {
 ViewBag.data = new Countries().CountriesList();
 model.Value = model.Value;
 return View(model);
 }
 }
}
```

### Data Annotation

Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end users.

Data Annotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET Framework will automatically enforce these validation rules and display validation messages in the view.

Using **value** property gets or sets the value of the selected item in the control.

### C#HTML

```
@using (Html.BeginForm())
{
 @Html.EJS().DropDownListFor(model =>
model.EnquiringAboutSelect).Placeholder("Select a
Country").DataSource(Model?.EnquiringAboutSelectListItems).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListBoxFieldSettings { Value = "Text"
}).Placeholder("Please Select Enquiring About").Render()
 <div>
 @Html.ValidationMessageFor(model => model.EnquiringAboutSelect)
 </div>
 <div id="submitButton">
 @Html.EJS().Button("btn").Content("Post").Render()
 </div>
}
```

### FOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public class AutoCompleteController : Controller
 {
 public ActionResult Index()
 {
 DataModel model = new DataModel();
 model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
 model.EnquiringAboutSelect = "104";
 return View(model);
 }
 [HttpPost]
 public ActionResult Index(DataModel model)
 {
 model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
 model.EnquiringAboutSelect = model.EnquiringAboutSelect;
 return View(model);
 }
 }
 public class DataModel
 {
 [Required(ErrorMessage = "The value is Required")]
 public string EnquiringAboutSelect { get; set; }
 }
}
```

```

 public List<ListItems> EnquiringAboutSelectListItems { get; set; }
 }
 public class ListItems
 {
 public string Text { get; set; }
 public string Value { get; set; }
 public List<ListItems> getListItems()
 {
 List<ListItems> items = new List<ListItems>();
 items.Add(new ListItems() { Text = "Aberdeen", Value = "103" });
 items.Add(new ListItems() { Text = "Alexandria", Value = "102"
 });
 items.Add(new ListItems() { Text = "Albany", Value = "101" });
 items.Add(new ListItems() { Text = "Beacon ", Value = "104" });
 items.Add(new ListItems() { Text = "Brisbane ", Value = "105"
 });
 return items;
 }
 }
}

```

## Migration from Essential JS 1

This article describes the API migration process of DropDownList component from Essential JS 1 to Essential JS 2.

### DataBinding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *Datasource*

<br/>@Html.EJ().DropDownList("countryList").Datasource((IEnumerable<Countries>)ViewBag.datasource1) | **Property:** *dataSource*

<br/>@Html.EJS().DropDownList("games").DataSource((IEnumerable<object>)ViewBag.localdata).Render() |

| **Fields for mapping** | **Property:** *DropDownListFields*

<br/>@Html.EJ().DropDownList("groupsList").DropDownListFields(f => f.Value("parentId").Text("text")) | **Property:** *Fields*

<br/>@Html.EJS().DropDownList("games").Fields(new DropDownListFieldSettings { Text = "Game" }).Render() |

| **Query** | **Property:** *Query*

<br/>@Html.EJ().DropDownList("groupsList").Query("ej.Query().from('Customers').take(6)") | **Property:**

*Query*<br/>@Html.EJS().DropDownList("games").Query((string)ViewBag.query).Render() |

| **Begin event** | **Event** :*ActionBegin*

<br/>@Html.EJ().DropDownList("selectCompany").ActionBegin("onBegin") | **Event** : *actionBegin*

<br/>@Html.EJS().DropDownList("customers").ActionBegin("onBegin").Render() |

| **Complete event** | **Event:** *ActionComplete*

<br/>@Html.EJ().DropDownList("selectCompany").ActionComplete("onBegin") | **Event:**

*ActionComplete*

```

@Html.EJS().DropDownList("customers").ActionComplete("ActionComplete").Render() |
```

| **Failure event** | **Event:** *ActionFailure*

```

@Html.EJ().DropDownList("selectCompany").ActionFailure("onFailure") | Event:
ActionFailure
```

```

@Html.EJS().DropDownList("customers").ActionFailure("ActionFailure").Render() |
```

| **Success event** | **Event:** *ActionSuccess*

```

@Html.EJ().DropDownList("selectCompany").ActionSuccess("onsuccess") | Not Applicable |
```

| **Data binding event** | **Event:** *DataBound* <br/>

```
@Html.EJ().DropDownList("selectCompany").DataBound("onBound") | Event: DataBound
```

```

@Html.EJS().DropDownList("customers").DataBound("onBound").Render() |
```

*Filtering*

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *EnableFilterSearch*

```

@Html.EJ().DropDownList("selectCompany").EnableFilterSearch(true) | Property:
```

```
AllowFiltering
@Html.EJS().DropDownList("customers").AllowFiltering(true).Render() |
```

| **Server filtering** | **Property:** *EnableServerFiltering*

```

@Html.EJ().DropDownList("selectCompany").EnableServerFiltering(true) | Property:
```

```
AllowFiltering
@Html.EJS().DropDownList("customers").AllowFiltering(true).Render() |
```

| **Filter type** | **Property:** *FilterType*

```

@Html.EJ().DropDownList("selectCompany").FilterType(SearchFilterType.StartsWith) |
```

<https://ej2.syncfusion.com/aspnetmvc/DropDownList/Filtering#/material> |

| **No Records Template** | **Not Applicable** | **Property:** *NoRecordsTemplate* <br/>

```
@Html.EJS().DropDownList("games").NoRecordsTemplate(" NO DATA
AVAILABLE").Render() |
```

| **Filter Bar watermarktext** | **Not Applicable** | **Property:** *FilterBarPlaceholder*

```

@Html.EJS().DropDownList("customers").FilterBarPlaceholder(true).Render() |
```

| **Ignore casing and diacritics** | **Not Applicable** | **Property:**

```
IgnoreAccent
@Html.EJS().DropDownList("customers").IgnoreAccent(true).Render() |
```

| **Incremental search** | **Property:**

```
EnableIncrementalSearch
@Html.EJ().DropDownList("selectCompany").EnableIncrementalSe
arch(true) | By default it is true |
```

| **Case sensitivity** | **Property:**

```
CaseSensitiveSearch
@Html.EJ().DropDownList("selectCompany").CaseSensitiveSearch(true)
```

<https://ej2.syncfusion.com/aspnetmvc/DropDownList/Filtering#/material> |

| **Search event** | **Event:** *Search*

```

@Html.EJ().DropDownList("selectCompany").Search("onSearch") | Event: Filtering
```

```

@Html.EJS().DropDownList("customers").Filtering("filtering").Render() |
```

## Template

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *Template*

```

@Html.EJ().DropDownList("selectCompany").Template("<div><img class='imgId'
src='../Content/Employees/${Image}.png' alt='employee'/> <div class='ename'> ${Text}
</div><div class='role'> ${Role} </div><div class='cont'> ${Country} </div></div>") | Property:
ItemTemplate
@Html.EJS().DropDownList("customers").ItemTemplate("@Html.Raw("${FirstName}<span class
='city'>${City}")").Render() |
```

| **Group Template** | **Not Applicable** | **Property:** *GroupTemplate*

```

@Html.EJS().DropDownList("customers").GroupTemplate("@Html.Raw("${City}</
strong>")").Render() |
```

| **ValueTemplate** | **Not Applicable** | **Property:** *ValueTemplate*

```

@Html.EJS().DropDownList("customers").ValueTemplate("@"@Html.Raw("${FirstNa
me} - ${City}")").Render() |
```

| **Header Template** | **Property:** *HeaderTemplate*

```

@Html.EJ().DropDownList("selectCompany").HeaderTemplate("<div
class='eheader'>PHOTO DETAILS</div>") | Property:
HeaderTemplate

@Html.EJS().DropDownList("customers").HeaderTemplate("@Html.Raw("NameCity")").Render() |
```

| **FooterTemplate** | **Not applicable** | **Property:** *FooterTemplate*

```

@Html.EJS().DropDownList("customers").FooterTemplate("@Html.Raw(" Total list items: " + 8 + "")").Render() |
```

| **No records Template** | **Not applicable** | **Property:** *NoRecordsTemplate*

```

@Html.EJS().DropDownList("customers").NoRecordsTemplate("@Html.Raw(" NO DATA AVAILABLE")").Render() |
```

| **Action failure Template** | **Not applicable** | **Property:** *ActionFailureTemplate*

```

@Html.EJS().DropDownList("customers").ActionFailureTemplate("@Html.Raw(" Data fetch get fails")").Render() |
```

## Virtual Scrolling

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *AllowVirtualScrolling*

```

@Html.EJ().DropDownList("customers").AllowVirtualScrolling(true) | Not applicable |
```

## Applying CSS

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *CssClass*

<br/>@Html.EJ().DropDownList("customers").CssClass("customClass") | **Property:** *CssClass*

<br/>@Html.EJS().DropDownList("customers").CssClass("class").Render() |

| **showRoundedCorner** | **Property:** *ShowRoundedCorner*

<br/>@Html.EJ().DropDownList("customers").ShowRoundedCorner(true) | **Property:** *CssClass*

<br/>@Html.EJS().DropDownList("customers").CssClass("class").Render() |

## Sorting

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *EnableSorting*

<br/>@Html.EJ().DropDownList("customers").EnableSorting(true) | **Enabled only on using sortorder** Property |

| **Order of sorting** | **Property:** *SortOrder*

<br/>@Html.EJ().DropDownList("customers").SortOrder("SortOrder.Descending") | **Property:**

*SortOrder* <br/>@Html.EJS().DropDownList("customers").SortOrder("Ascending").Render() |

## Popup

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Popup height** | **Property:** *PopupHeight*

<br/>@Html.EJ().DropDownList("customers").PopupHeight("550px") | **Property:** *PopupHeight*

<br/>@Html.EJS().DropDownList("customers").PopupHeight("220px").Render() |

| **Popup width** | **Property:** *PopupWidth*

<br/>@Html.EJ().DropDownList("customers").PopupWidth("550px") | **Property:** *PopupWidth*

<br/>@Html.EJS().DropDownList("customers").PopupWidth("300px").Render() |

| **Popup show on load** | **Property:** *ShowPopupOnLoad* <br/>

@Html.EJ().DropDownList("customers").ShowPopupOnLoad(true) | **By default, the data load on demand.** |

| **enableAnimation** | **Property:** *EnableAnimation*

<br/>@Html.EJ().DropDownList("customers").EnableAnimation(true) | **Not applicable** |

| **Popup resizing** | **Property:** *EnablePopupResize*

<br/>@Html.EJ().DropDownList("customers").EnablePopupResize(true) | **Not applicable** |

| **Maximum Popup height** | **Property:** *MaxPopupHeight*

<br/>@Html.EJ().DropDownList("customers").MaxPopupHeight("550px") | **Not applicable** |

| **Minimum Popup height** | **Property:** *min-popup-*

*height* <br/>@Html.EJ().DropDownList("customers").MinPopupHeight("550px")<br/>}); | **Not applicable** |

| **Maximum Popup width** | **Property:** *MaxPopupWidth*

<br/>@Html.EJ().DropDownList("customers").MaxPopupWidth("550px") | **Not applicable** |

| **Minimum Popup width** | **Property:** *MinPopupWidth*

<br/>@Html.EJ().DropDownList("customers").MinPopupWidth("550px") | **Not applicable** |

| **Loading data** | **Property:** *LoadOnDemand*

<br/>@Html.EJ().DropDownList("customers").LoadOnDemand(true) | **By default, it is true** |

| **Popup showing manually** | **Method:** *showPopup*

<br/>@Html.EJ().DropDownList("dropdown")<br/><br/>\$('#dropdown').ejDropDownList('showPopup') | **Method:** *showPopup*

<br/>@Html.EJS().DropDownList("dropdown").Render()<br/><br/>var ddlObj = document.getElementById('dropdown').ej2\_Instances[0];<br/><br/>ddlObj.showPopup(); |

| **Popup hiding manually** | **Method:** *hidePopup*

<br/>@Html.EJ().DropDownList("dropdown")<br/><br/>\$('#dropdown').ejDropDownList('hidePopup') | **Method:**

*HidePopup*<br/>@Html.EJS().DropDownList("dropdownlist").Render()<br/><br/>var ddlObj = document.getElementById("dropdownlist").ej2\_Instances[0];<br/><br/>ddlObj.hidePopup(); |

| **Before Popup hide event** | **Event:** *BeforePopupHide*

<br/>@Html.EJ().DropDownList("dropdown").BeforePopupHide("event") | **Not applicable** |

| **Before Popup shown event** | **Event:**

*BeforePopupShown*<br/>@Html.EJ().DropDownList("dropdown").BeforePopupShown("event") |

**Event:** *BeforeOpen* <br/>@Html.EJS().DropDownList("dropdown").BeforeOpen("event").Render() |

| **Popup hide event** | **Event:**

*PopupHide*<br/>@Html.EJ().DropDownList("dropdown").PopupHide("event") | **Event:** *Close*

<br/>@Html.EJS().DropDownList("dropdown").Close("event").Render() |

| **Popup resize event** | **Event:**

*PopupResize*<br/>@Html.EJ().DropDownList("dropdown").PopupResize("event") | **Not applicable**

|

| **Popup resize start event** | **Event:**

*PopupResizeStart*<br/>@Html.EJ().DropDownList("dropdown").PopupResizeStart("event") | **Not applicable** |

| **Popup resize stop event** | **Event:**

*popupResizeStop*<br/>@Html.EJ().DropDownList("dropdown").PopupResizeStop("event") | **Not applicable** |

| **Popup shown event** | **Event:**

*Popupshown*<br/>@Html.EJ().DropDownList("dropdown").PopupShown("event") | **Event:**

*Open*<br/>@Html.EJS().DropDownList("dropdown").Open("event").Render() |

[Placeholder](#)

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|



| **Watermark text** | **Property:** *WatermarkText*  
 <br/>@Html.EJ().DropDownList("dropdown").WatermarkText("Select") | **Property:** *Placeholder*  
 <br/>@Html.EJS().DropDownList("dropdown").Placeholder("Select").Render() |

| **Floating of waterMarkText** | **Not applicable** | **Property:** *FloatLabelType*  
 <br/>@Html.EJS().DropDownList("dropdown").FloatLabelType("Auto").Render() |

### Grouping

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Default** | **Property:** *fields.groupBy*  
 <br/>@Html.EJ().DropDownList("groupsList").DropDownListFields(f => f.GroupBy("parentId")) |  
**Property:** *fields.groupBy*<br/>@Html.EJS().DropDownList("games").Fields(new  
 DropDownListFieldSettings { GroupBy = "Game" }).Render() |

| **Group Template** | **Not applicable** | **Property:**  
*GroupTemplate*<br/>@Html.EJS().DropDownList("customers").GroupTemplate("@Html.Raw("<strong>\${City}</strong>")").Render() |

### Accessibility

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Globalization** | **Property:** *Locale*<br/>@Html.EJ().DropDownList("customers").Locale("fr-FE") |  
**Property:** *Locale*<br/>@Html.EJS().DropDownList("customers").Locale("fr-FE").Render() |

| **Rtl support** | **Property:** *EnableRtl*<br/>@Html.EJ().DropDownList("customers").EnableRtl(true) |  
**Property:** *EnableRtl*<br/>@Html.EJS().DropDownList("customers").EnableRtl(true).Render() |

### Miscellaneous

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Enable/disable** | **Property:** *Enabled*<br/>@Html.EJ().DropDownList("customers").Enabled(true) |  
**Property:** *Enabled* <br/>@Html.EJS().DropDownList("customers").Enabled(true).Render() |

| **Read only** | **Property:** *ReadOnly* <br/>@Html.EJ().DropDownList("customers").ReadOnly(true) |  
 <br/>**Property:** *ReadOnly*<br/>@Html.EJS().DropDownList("customers").ReadOnly(true).Render()  
 |

| **Persistence of data** | **Property:**  
*EnablePersistence*<br/>@Html.EJ().DropDownList("customers").EnablePersistence(true)  
**Property:**  
*EnablePersistence*<br/>@Html.EJS().DropDownList("customers").EnablePersistence(true).Render()  
 ) |

| **Disable** | **Method:** *disable*<br/>@Html.EJ().DropDownList("dropdown")<br/>  
 <br/>\$('#dropdown').ejDropDownList('disable') | **Property:**  
*Enabled*<br/>@Html.EJS().DropDownList("customers").Enabled(false).Render() |

| **Enable** | **Method:** `enable`<br/>@Html.EJ().DropDownList("dropdown")<br/>

<br/>\$('#dropdown').ejDropDownList('enable') | **Property:**

`Enabled`<br/>@Html.EJS().DropDownList("customers").Enabled(true).Render() |

| **Height** | **Property:** `Height` <br/>@Html.EJ().DropDownList("dropdown").Height("300px") |

**Property:** `Height` <br/>@Html.EJS().DropDownList("customers").Height("300px").Render() |

| **Width** | **Property:** `Width` <br/>@Html.EJ().DropDownList("dropdown").Width("300px") |

**Property:** `Width` <br/>@Html.EJS().DropDownList("customers").Width("300px").Render() |

## Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Selecting particular index** | **Property:** `selected-`

`index`<br/>@Html.EJ().DropDownList("dropdown").Index(3) | **Property:**

`index`<br/>@Html.EJS().DropDownList("customers").Index(3).Render() |

| **Selecting particular value** | **Property:** `value`<br/>@Html.EJ().DropDownList("dropdown").Value(3)

| **Property:** `value`<br/>@Html.EJS().DropDownList("customers").Value(3).Render() |

| **Selecting particular text** | **Property:** `text`

<br/>@Html.EJ().DropDownList("dropdown").Text("data") | **Property:**

`text`<br/>@Html.EJS().DropDownList("customers").Text("data").Render() |

| **Target id** | **Property:** `targetId` <br/>@Html.EJ().DropDownList("dropdown").TargetId("data") |

**Not applicable** |

| **Selecting item using text** | **Method:** `selectItemByText`

<br/>@Html.EJ().DropDownList("dropdown")<br/>

<br/>\$('#dropdown').ejDropDownList('selectItemByText','car') | **Not applicable** |

| **Unselect item using text** | **Method:**

`unselectItemByText`<br/>@Html.EJ().DropDownList("dropdown")<br/>

<br/>\$('#dropdown').ejDropDownList('unselectItemByText','car') | **Not applicable** |

| **Selecting item using value** | **Method:**

`selectItemByValue`<br/>@Html.EJ().DropDownList("dropdown")<br/>

<br/>\$('#dropdown').ejDropDownList('selectItemByValue','car') | **Not applicable** |

| **Getting data by using value** | **Method:**

`getItemDataByValue`<br/>@Html.EJ().DropDownList("dropdown")<br/><br/>\$('#dropdown').ejDr  
opDownList('unselectItemByValue','car') | **Method:**

`getDataByValue`<br/>@Html.EJS().DropDownList("dropdownlist").Render()<br/><br/>var ddlObj =  
document.getElementById("dropdownlist").ej2\_Instances[0];<br/><br/>ddlObj.getItemDataByV  
alue("data"); |

| **Get selected value** | **Method:** `getSelectedItem`<br/>@Html.EJ().DropDownList("dropdown")<br/>

<br/>\$('#dropdown').ejDropDownList('getSelectedItem') | **Not applicable** |

| **Get selected text** | **Method:** `getSelectedText<br/>@Html.EJ().DropDownList("dropdown")<br/><br/>$('#dropdown').ejDropDownList('getSelectedText')` | **Property:** `text<br/>@Html.EJS().DropDownList("customers").Text("data").Render()` |

| **Select event** | **Event:** `select<br/>@Html.EJ().DropDownList("dropdown").select("onSelect")` | **Event:** `select<br/>@Html.EJS().DropDownList("customers").Select("onSelect").Render()` |

| **Addition of Html attributes** | **Property:** `html-attributes<br/>@Html.EJS().DropDownList("customers").HtmlAttributes((IDictionary<string,object>)ViewData["HtmlAttrData"])` | **Property:** `htmlAttributes<br/>@Html.EJS().DropDownList("customers").HtmlAttributes(@ViewBag.data).Render()` |

### Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| **Adding new item** | **Method :** `addItem<br/><ej-drop-down-list></ej-drop-down-list><br/><br/>$('#dropdown').ejDropDownList("addItem", {text:"India"});` | **Method:** `addItem<br/><ejs-dropdownlist><br/><br/>var ddlObj = document.getElementById(dropdownlist).ej2_Instances[0];<br/><br/>ddlObj.addItem({Id: 'game4', Game: 'Golf'}, 2);` |

| **Clearing the text** | **Method :** `clearText<br/><ej-drop-down-list></ej-drop-down-list><br/><br/>$('#dropdown').ejDropDownList('clearText')` | **Property:** `showClearButton<br/><ejs-dropdownlist showClearButton=true/>` |

| **Destroy the component** | **Method :** `destroy<br/><ej-drop-down-list></ej-drop-down-list><br/><br/>$('#dropdown').ejDropDownList('destroy')` | **Method:** `destroy<br/><ejs-dropdownlist><br/><br/>var ddlObj = document.getElementById(dropdownlist).ej2_Instances[0];<br/><br/>ddlObj.destroy;` |

| **Getting the data** | **Method :** `getListData<br/><ej-drop-down-list></ej-drop-down-list><br/><br/>$('#dropdown').ejDropDownList('getListData')` | **Method :** `getItems<br/><ejs-dropdownlist><br/><br/>var ddlObj = document.getElementById(dropdownlist).ej2_Instances[0];<br/><br/>ddlObj.getItems;` |

| **Create event** | **Event:** `create<br/><ej-drop-down-list create="onCreate"></ej-drop-down-list>` | **Event:** `created<br/><ejs-dropdownlist created ="created" />` |

| **Destroy event** | **Event:** `destroy<br/><ej-drop-down-list destroy="ondestroy"></ej-drop-down-list>` | **Event:** `destroyed<br/><ejs-dropdownlist destroyed ="destroy" />` |

| **Cascade event** | **Event:** `cascade<br/><ej-drop-down-list cascade="cascade"></ej-drop-down-list>` | <https://ej2.syncfusion.com/demos/#/material/drop-down-list/cascading.html> |

| **Change event** | **Event:** `change<br/><ej-drop-down-list create="change"></ej-drop-down-list>` | **Event:** `change<br/><ejs-dropdownlist change ="change" />` |

| **Focus out event** | **Event:** `focusOut<br/><ej-drop-down-list focus-out="focus"></ej-drop-down-list>` | **Event:** `blur<br/><ejs-dropdownlist blur ="blur" />` |

| **Focus in event** | **Event:** `focusIn<br/><br/><ej-drop-down-list focus-in="focus"></ej-drop-down-list>` | **Event:** `focus<br/><ejs-dropdownlist focus="onfocus" />` |

## DropDownTree

### Getting Started with ASP.NET MVC DropDownTree Control

This section briefly explains about how to include [ASP.NET MVC DropDownTree](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### **~/ LAYOUT.CSHTML**

```
<head>
...
```

```
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

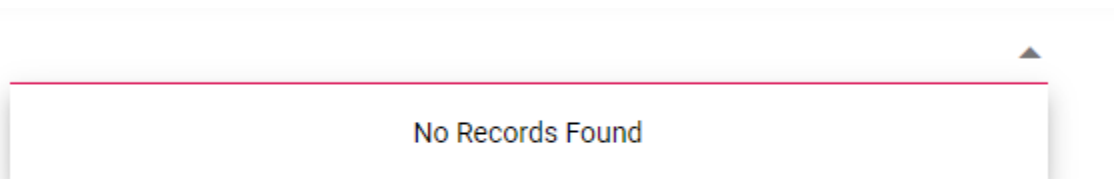
### Add ASP.NET MVC DropDownTree control

Now, add the Syncfusion ASP.NET MVC DropDownTree control in `~/Views/Home/Index.cshtml` page.

### CSHTML

```
@Html.EJS().DropDownTree("data").Render()
```

Press `Ctrl+F5` (Windows) or `⌘+F5` (macOS) to run the app. Then, the Syncfusion ASP.NET MVC DropDownTree control will be rendered in the default web browser.



### Binding data source

The Dropdown Tree control can load data either from local data sources or remote data services. This can be done using the `dataSource` property that is a member of the [Fields](#) property. The `dataSource` property supports a nested list of data and `DataManager`. Here, a nested list of data is passed to the Dropdown Tree control.

### CSHTML

```
@model List<DropDownTreeSample.Controllers.Parent>
```

```
@{

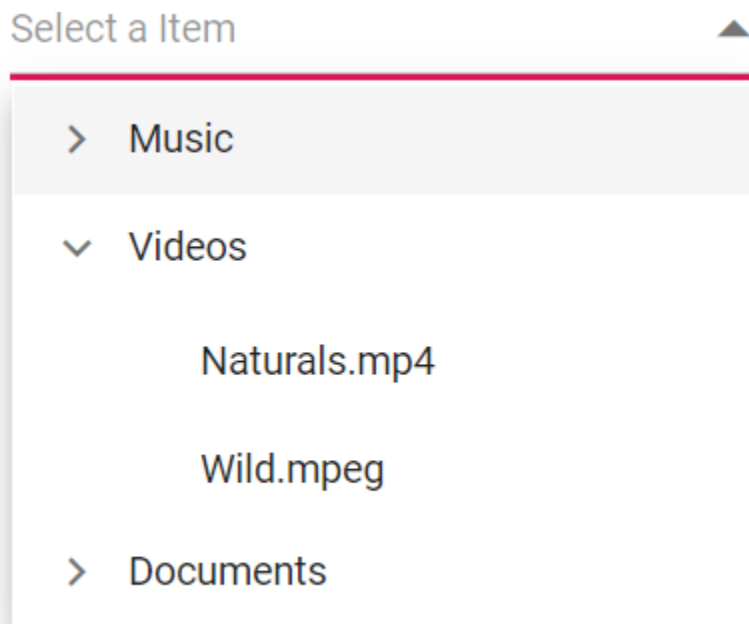
 char[] value = { 'n', 'o', 'd', 'e', 'C', 'h', 'i', 'l', 'd' };
 string child = new string(value);
}
<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").Placeholder("Select a
 Item").Fields(field =>

 field.Value("nodeId").Child(child).Text("nodeText").DataSource(Model)).Rende
 r()
</div>
```

### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 List<Parent> parentitem = new List<Parent>();
 List<Child> childitem = new List<Child>();
 parentitem.Add(new Parent
 {
 nodeId = "01",
 nodeText = "Music",
 nodeChild = childitem,
 });
 childitem.Add(new Child { nodeId = "01-01", nodeText = "Gouttes.mp3" });
 List<Child> childitem2 = new List<Child>();
 parentitem.Add(new Parent
 {
 nodeId = "02",
 nodeText = "Videos",
 expanded = true,
 nodeChild = childitem2,
 });
 childitem2.Add(new Child { nodeId = "02-01", nodeText = "Naturals.mp4" });
 childitem2.Add(new Child { nodeId = "02-02", nodeText = "Wild.mpeg" });
 List<Child> childitem3 = new List<Child>();
 parentitem.Add(new Parent
 {
 nodeId = "03",
 nodeText = "Documents",
 nodeChild = childitem3,
 });
 childitem3.Add(new Child { nodeId = "03-01", nodeText = "Environment
 Pollution.docx" });
 childitem3.Add(new Child { nodeId = "03-02", nodeText = "Global Water,
 Sanitation, & Hygiene.docx" });
 childitem3.Add(new Child { nodeId = "03-03", nodeText = "Global Warming.ppt"
 });
 childitem3.Add(new Child { nodeId = "03-04", nodeText = "Social Network.pdf"
 });
 childitem3.Add(new Child { nodeId = "03-05", nodeText = "Youth
 Empowerment.pdf" });
 //ViewBag.dataSource = parentitem;
 }
}
```

```
return View(parentitem);
}
....
}
public class Parent
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<Child> nodeChild;
}
public class Child
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<Child> nodeChild;
}
```



**Note:** [View Sample in GitHub.](#)

### Data Binding in Dropdown Tree Component

The Dropdown Tree control provides an option to load the data either from local data sources or from remote data services. This can be done through `dataSource` property that is a member of the `fields` property. The `dataSource` property supports list of data and `DataManager`. It also supports different

kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

Dropdown Tree has **load on demand** (Lazy load) option. It reduces the bandwidth size while consuming the huge data. By default, the **loadOnDemand** is set to false. By enabling this property, it loads first level items initially, and when parent item is expanded, it loads the child items based on the **parentValue/child** member.

#### Local data

To bind local data to the Dropdown Tree, you can assign a list of data to the **dataSource** property.

The Dropdown Tree control requires three fields (Value, text, and parentValue) to render local data source. When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be provided as an instance of the **DataManager**. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

#### Hierarchical data

Dropdown Tree can be populated with the hierarchical data source that contains nested list of data. You can directly map the hierarchical data and the field members with corresponding key values from the hierarchical data to the **fields** property.

In the following example, **code**, **name**, and **countries** columns from hierarchical data have been mapped to **value**, **text**, and **child** fields, respectively.

#### CSHTML

```
<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").Fields(field =>

 field.Value("code").Child(ViewBag.child).Text("name").DataSource(ViewBag.data)).Render()
</div>
```

#### HIERARCHICALDATA.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownTree.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownTree.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult HierarchicalData()
 {
 List<Continents> continents = new List<Continents>();
 List<Countries> countries = new List<Countries>();
```



```

 continents.Add(new Continents
 {
 code = "NA",
 name = "North America",
 expanded = true,
 child = countries,
 });
 countries.Add(new Countries { code = "USA", name = "United
States of America", selected = true });
 countries.Add(new Countries { code = "CUB", name = "Cuba" });
 countries.Add(new Countries { code = "MEX", name = "Mexico" });
 List<Countries> countries2 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "AF",
 name = "Africa",
 child = countries2,
 });
 countries2.Add(new Countries { code = "NGA", name = "Nygeria"
});
 countries2.Add(new Countries { code = "EGY", name = "Egypt" });
 countries2.Add(new Countries { code = "ZAF", name = "South
Africa" });
 List<Countries> countries3 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "AS",
 name = "Asia",
 child = countries3,
 });
 countries3.Add(new Countries { code = "CHN", name = "China" });
 countries3.Add(new Countries { code = "IND", name = "India" });
 countries3.Add(new Countries { code = "JPN", name = "Japan" });
 List<Countries> countries4 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "EU",
 name = "Europe",
 child = countries4,
 });
 countries4.Add(new Countries { code = "DNK", name = "Denmark"
});
 countries4.Add(new Countries { code = "FIN", name = "Finland"
});
 countries4.Add(new Countries { code = "AUT", name = "Austria"
});
 List<Countries> countries5 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "SA",
 name = "South America",
 child = countries5,
 });
 countries5.Add(new Countries { code = "BRA", name = "Brazil" });
 countries5.Add(new Countries { code = "COL", name = "Colombia"
});

```

```

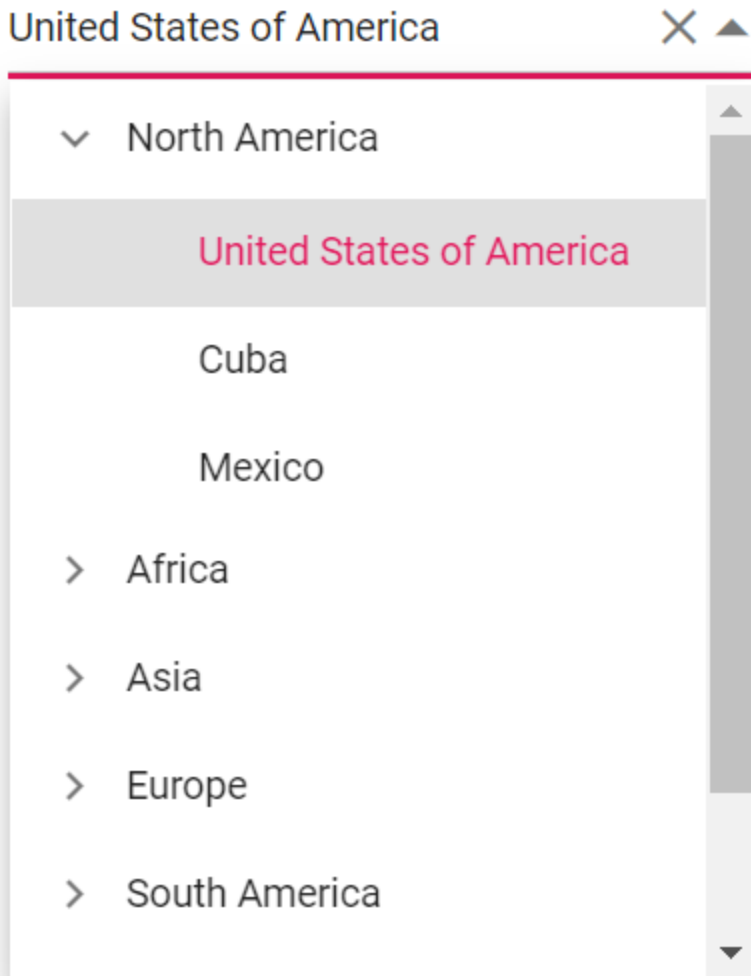
 countries5.Add(new Countries { code = "ARG", name = "Argentina"
 });

 List<Countries> countries6 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "OC",
 name = "Oceania",
 child = countries6,
 });
 countries6.Add(new Countries { code = "AUS", name = "Australia"
 });
 countries6.Add(new Countries { code = "NZL", name = "Newzealand"
 });
 countries6.Add(new Countries { code = "WSM", name = "Samoa" });
 List<Countries> countries7 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "AN",
 name = "Antartica",
 child = countries7,
 });
 countries7.Add(new Countries { code = "BVT", name = "Bouvet
Island" });
 countries7.Add(new Countries { code = "ATF", name = "French
Southern Lands" });
 char[] value = { 'c', 'h', 'i', 'l', 'd' };
 string Child = new string(value);
 ViewBag.child = Child;
 ViewBag.data = continents;
 return View();
}

public class Continents
{
 public string code;
 public string name;
 public bool expanded;
 public bool selected;
 public List<Countries> child;
}

public class Countries
{
 public string code;
 public string name;
 public bool expanded;
 public bool selected;
}
}

```



#### Self-referential data

DropDown Tree can be populated from the self-referential data structure that contains array of JSON objects with `parentValue` mapping.

You can directly assign self-referential data and map all the field members with corresponding key values from self-referential data to the `fields` property.

To render the root level items, specify the `parentValue` as null or no need to specify the `parentValue` in the `dataSource`.

In the following example, `id`, `pid`, `hasChild`, and `name` columns from self-referential data have been mapped to `value`, `parentValue`, `hasChildren`, and `text` fields, respectively.

#### CSHTML

```
<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").Fields(field =>

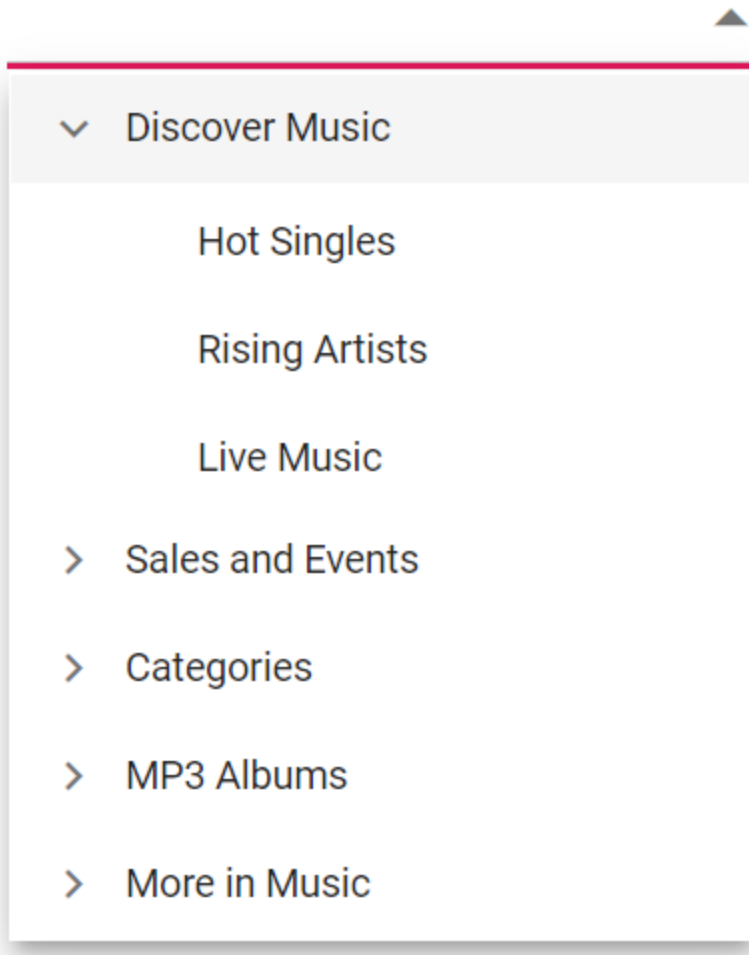
 field.Value("id").ParentValue("pid").HasChildren("hasChild").Text("name").DataSource(ViewBag.dataSource).Render()
</div>
```

**LISTDATA.CS**

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownTree.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownTree.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult ListData()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Live Music"
 });
 treedata.Add(new
 {
 id = 5,
 hasChild = true,
 name = "Sales and Events",
 });
 treedata.Add(new
 {
 id = 6,
 pid = 5,
 name = "100 Albums - $5 Each",
 });
 treedata.Add(new
```

```
{
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
});
treedata.Add(new
{
 id = 8,
 pid = 5,
 name = "CD Deals"
});
treedata.Add(new
{
 id = 10,
 hasChild = true,
 name = "Categories"
});
treedata.Add(new
{
 id = 11,
 pid = 10,
 name = "Bestselling Albums",
});
treedata.Add(new
{
 id = 12,
 pid = 10,
 name = "New Releases"
});
treedata.Add(new
{
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
});
treedata.Add(new
{
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
});
treedata.Add(new
{
 id = 15,
 pid = 14,
 name = "Rock"
});
treedata.Add(new
{
 id = 16,
 name = "Gospel",
 pid = 14,
});
treedata.Add(new
{
 id = 17,
 pid = 14,
```

```
 name = "Latin Music"
 });
 treedata.Add(new
 {
 id = 18,
 pid = 14,
 name = "Jazz"
 });
 treedata.Add(new
 {
 id = 19,
 hasChild = true,
 name = "More in Music"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 19,
 name = "Music Trade-In"
 });
 treedata.Add(new
 {
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
 });
 treedata.Add(new
 {
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
 });
 ViewBag.dataSource = treedata;
 return View();
}
}
```



#### Remote data

Dropdown Tree can also be populated from a remote data service with the help of the **DataManager** control and **Query** property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

You can assign service data as an instance of **DataManager** to the **dataSource**. To interact with remote data source, you must provide the endpoint **url**.

The **DataManager** that acts as an interface between the service endpoint and the Dropdown Tree requires the following information to interact with service endpoint properly.

- **DataManager->url**: Defines the service endpoint to fetch data.
- **DataManager->adaptor**: Defines the adaptor option. By default, **ODataAdaptor** is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The **@syncfusion/ej2-data** package provides some pre-defined adaptors designed to interact with service endpoints. They are,

- **UrlAdaptor**: Used to interact with remote services. This is the base adaptor for all remote based adaptors.
- **ODataAdaptor**: Used to interact with OData endpoints.
- **ODataV4Adaptor**: Used to interact with OData V4 endpoints.
- **WebApiAdaptor**: Used to interact with Web API created under OData standards.
- **WebMethodAdaptor**: Used to interact with web methods.

In the following example, **ODataV4Adaptor** is used to fetch data from the remote services. The **EmployeeID**, **FirstName**, and **EmployeeID** columns from the Employees table has been mapped to **value**, **text**, and **hasChildren** fields respectively for first level nodes.

The **OrderID**, **EmployeeID**, and **ShipName** columns from the orders table have been mapped to **value**, **parentValue**, and **text** fields respectively for second level nodes.

### CSHTML

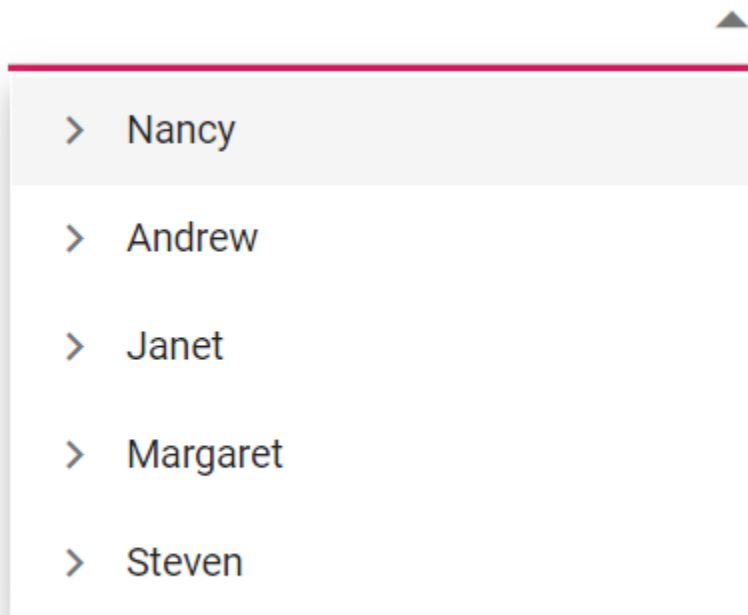
```
<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("tree").Fields(field =>
 field.Query("new
ej.data.Query().from('Employees').select('EmployeeID,FirstName,Title').take(
5)").Value("EmployeeID").ParentValue("pid").Selected("selected")
.Text("FirstName").HasChildren("EmployeeID")
.DataSource(dataSource =>
{
dataSource.Url("https://services.odata.org/V4/Northwind/Northwind.svc").Adap
tor("ODataV4Adaptor").CrossDomain(true);
}) .Child(ViewBag.child) .Render()
})
</div>
```

### REMOTEDATA.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownTree.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownTree.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult RemoteData()
 {
 DropDownTreeFields childData = new DropDownTreeFields();
 childData.Query = "new
ej.data.Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5)
";
 childData.Value = "OrderID";
 childData.Text = "ShipName";
 childData.ParentValue = "EmployeeID";
 childData.DataSource = new Syncfusion.EJ2.DataManager
```



```
{
 Url =
 "https://services.odata.org/V4/Northwind/Northwind.svc",
 Adaptor = "ODataV4Adaptor",
 CrossDomain = true
};
ViewBag.child = childData;
return View();
}
}
```



### Prevent Node selection

You can prevent the selection of individual tree node by using the **Selectable** property. The tree node selection is not allowed while disable this property.

The **Selectable** property is disabled and the selection is prevented for parent nodes in below sample.

### CSHTML

```
<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").Fields(field =>

 field.Value("id").ParentValue("pid").HasChildren("hasChild").Text("name").Se
 lectable("selectable").DataSource(ViewBag.dataSource)).Render()
</div>
```

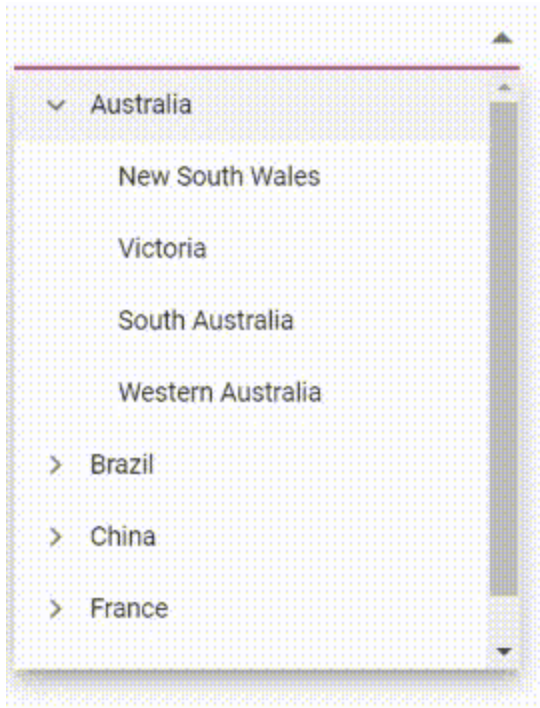
### PREVENTNODE.CS

```
using System;
using System.Collections.Generic;
```

```
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownTree.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownTree.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult ListData()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true,
 selectable = false
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Live Music"
 });
 treedata.Add(new
 {
 id = 5,
 hasChild = true,
 name = "Sales and Events",
 selectable = false
 });
 treedata.Add(new
 {
 id = 6,
 pid = 5,
 name = "100 Albums - $5 Each",
 });
 treedata.Add(new
 {
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
 });
 }
 }
}
```

```
});
treedata.Add(new
{
 id = 8,
 pid = 5,
 name = "CD Deals"
});
treedata.Add(new
{
 id = 10,
 hasChild = true,
 name = "Categories",
 selectable = false
});
treedata.Add(new
{
 id = 11,
 pid = 10,
 name = "Bestselling Albums",
});
treedata.Add(new
{
 id = 12,
 pid = 10,
 name = "New Releases"
});
treedata.Add(new
{
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
});
treedata.Add(new
{
 id = 14,
 hasChild = true,
 name = "MP3 Albums",
 selectable = false
});
treedata.Add(new
{
 id = 15,
 pid = 14,
 name = "Rock"
});
treedata.Add(new
{
 id = 16,
 name = "Gospel",
 pid = 14,
});
treedata.Add(new
{
 id = 17,
 pid = 14,
 name = "Latin Music"
});
```

```
treedata.Add(new
{
 id = 18,
 pid = 14,
 name = "Jazz"
});
treedata.Add(new
{
 id = 19,
 hasChild = true,
 name = "More in Music",
 selectable = false
});
treedata.Add(new
{
 id = 20,
 pid = 19,
 name = "Music Trade-In"
});
treedata.Add(new
{
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
});
treedata.Add(new
{
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
});
ViewBag.dataSource = treedata;
return View();
}
}
```



### Templates in Drop Down Tree Component

The Dropdown Tree provides support to customize each list item, header, and footer elements. It uses the Essential JS 2 [Template engine](#) to compile and render the elements properly.

#### Item template

The content of each list item within the Dropdown Tree can be customized with the help of `itemTemplate` property.

In the following sample, the Dropdown Tree list items are customized with employee information such as **name** and **job** using the `itemTemplate` property.

The template expression should be provided inside the `${...}` interpolation syntax.

#### CSHTML

```
<div class="control-section">
 <div style="margin: 0 auto; width:350px;">

@Html.EJS().DropDownTree("ddt").CssClass("custom").Placeholder("Select an
employee").Width("350px").PopupHeight("250px").Fields(ViewBag.templateData).
ItemTemplate("<div>${name} - ${job}</div>").Render()
 </div>
</div>
<style>
 .custom .ejob {
 opacity: .60;
 }
</style>
```

#### ITEMTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields templateData = new DropDownTreeFields();
 public ActionResult Index()
 {
 DropDownTreeTemplate dropdownTreeTemplate = new
DropDownTreeTemplate();
 templateData.DataSource = dropdownTreeTemplate.Template();
 templateData.Value = "id";
 templateData.Text = "name";
 templateData.Expanded = "expanded";
 templateData.HasChildren = "hasChild";
 templateData.ParentValue = "pid";
 ViewBag.templateData = templateData;
 ViewBag.localData = templateData.DataSource;
 return View();
 }
 }
}

```

### DROPDOWNTREETEMPLATE.CS

```

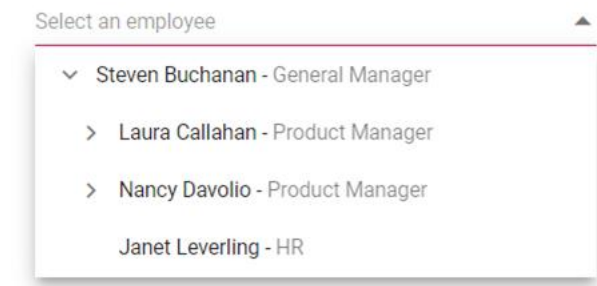
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 localData.Add(new { id = 1, name = "Steven Buchanan", hasChild =
true, expanded = true, job = "General Manager" });
 localData.Add(new { id = 2, pid = 1, name = "Laura Callahan",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 3, pid = 2, name = "Andrew Fuller",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 4, pid = 3, name = "Anne Dodsworth",
job = "Developer" });
 localData.Add(new { id = 5, pid = 3, name = "Lilly", job =
"Developer", status = "online" });
 localData.Add(new { id = 6, pid = 1, name = "Nancy Davolio",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 7, pid = 6, name = "Michael Suyama",
job = "Team Lead", hasChild = true });
 }
 }
}

```

```

 localData.Add(new { id = 8, pid = 7, name = "Robert King", job
= "Developer" });
 localData.Add(new { id = 9, pid = 7, name = "Mary", job =
"Developer" });
 localData.Add(new { id = 10, pid = 1, name = "Janet Leverling",
job = "HR" });
 return localData;
 }
}

```



### Header template

The header element is shown statically at the top of the popup list items within the Dropdown Tree. A custom element can be placed as a header element using the `headerTemplate` property.

### CSHTML

```

<div class="control-section">
 <div style="margin: 0 auto; width:350px;">

@Html.EJS().DropDownTree("ddt").CssClass("custom").Placeholder("Select an
employee").Width("350px").PopupHeight("250px").Fields(ViewBag.templateData).
ItemTemplate("<div class=\"ename\">${name}</div>").HeaderTemplate("<div
class=\"head\">Employee List</div>").Render()
 </div>
</div>
<style>
 .custom .head {
 height: 40px;
 line-height: 40px;
 font-size: 14px;
 margin: 0 auto;
 width: 100%;
 padding: 0 20px;
 font-weight: bold;
 border-bottom: 1px solid #e0e0e0;
 }
</style>

```

### HEADERTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

```

```

using System.Web.Mvc;
using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields templateData = new DropDownTreeFields();
 public ActionResult Index()
 {
 DropDownTreeTemplate dropdownTreeTemplate = new
DropDownTreeTemplate();
 templateData.DataSource = dropdownTreeTemplate.Template();
 templateData.Value = "id";
 templateData.Text = "name";
 templateData.Expanded = "expanded";
 templateData.HasChildren = "hasChild";
 templateData.ParentValue = "pid";
 ViewBag.templateData = templateData;
 ViewBag.localData = templateData.DataSource;
 return View();
 }
 }
}

```

#### DROPDOWNTREETEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 localData.Add(new { id = 1, name = "Steven Buchanan", hasChild =
true, expanded = true, job = "General Manager" });
 localData.Add(new { id = 2, pid = 1, name = "Laura Callahan",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 3, pid = 2, name = "Andrew Fuller",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 4, pid = 3, name = "Anne Dodsworth",
job = "Developer" });
 localData.Add(new { id = 5, pid = 3, name = "Lilly", job =
"Developer", status = "online" });
 localData.Add(new { id = 6, pid = 1, name = "Nancy Davolio",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 7, pid = 6, name = "Michael Suyama",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 8, pid = 7, name = "Robert King", job
= "Developer" });
 localData.Add(new { id = 9, pid = 7, name = "Mary", job =
"Developer" });
 }
 }
}

```

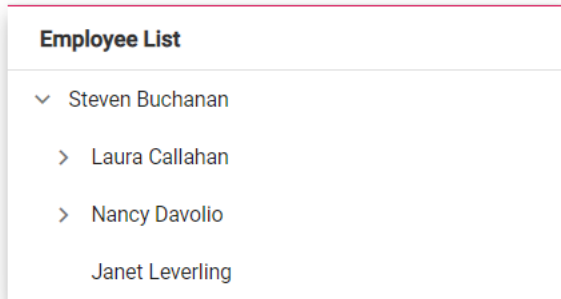


```

 localData.Add(new { id = 10, pid = 1, name = "Janet Leverling",
 job = "HR" });
 return localData;
 }
}

```

Select an employee



### Footer template

The DropDown Tree has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the `footerTemplate` property.

### CSHTML

```

<div class="control-section">
 <div style="margin: 0 auto; width:350px;">

@Html.EJS().DropDownTree("ddt").CssClass("custom").Placeholder("Select an
employee").Width("350px").PopupHeight("250px").Fields(ViewBag.templateData).
ItemTemplate("<div class=\"ename\">${name}</div>").FooterTemplate("<div
class=\"footer\"><div class=\"footer-content\">Total number of employees: 10
</div></div>").Render()
 </div>
</div>
<style>
 .custom .footer {
 height: 40px;
 line-height: 40px;
 font-size: 14px;
 margin: 0 auto;
 width: 100%;
 padding: 0 20px;
 font-weight: bold;
 }
 .custom .e-ddt-footer{
 border-top: 1px solid #e0e0e0;
 }
</style>

```

### FOOTERTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Web;
using System.Web.Mvc;
using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields templateData = new DropDownTreeFields();
 public ActionResult Index()
 {
 DropDownTreeTemplate dropdownTreeTemplate = new
DropDownTreeTemplate();
 templateData.DataSource = dropdownTreeTemplate.Template();
 templateData.Value = "id";
 templateData.Text = "name";
 templateData.Expanded = "expanded";
 templateData.HasChildren = "hasChild";
 templateData.ParentValue = "pid";
 ViewBag.templateData = templateData;
 ViewBag.localData = templateData.DataSource;
 return View();
 }
 }
}

```

#### **DROPDOWNTREETEMPLATE.CS**

```

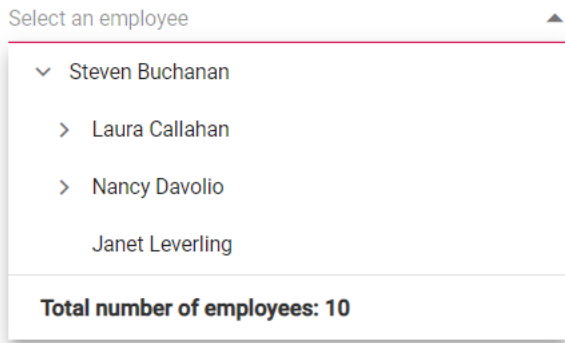
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 localData.Add(new { id = 1, name = "Steven Buchanan", hasChild =
true, expanded = true, job = "General Manager" });
 localData.Add(new { id = 2, pid = 1, name = "Laura Callahan",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 3, pid = 2, name = "Andrew Fuller",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 4, pid = 3, name = "Anne Dodsworth",
job = "Developer" });
 localData.Add(new { id = 5, pid = 3, name = "Lilly", job =
"Developer", status = "online" });
 localData.Add(new { id = 6, pid = 1, name = "Nancy Davolio",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 7, pid = 6, name = "Michael Suyama",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 8, pid = 7, name = "Robert King", job
= "Developer" });
 }
 }
}

```

```

 localData.Add(new { id = 9, pid = 7, name = "Mary", job =
"Developer" });
 localData.Add(new { id = 10, pid = 1, name = "Janet Leverling",
job = "HR" });
 return localData;
 }
}

```



### No records template

The DropDown Tree is supports to display custom design in the popup list content using the `noRecordsTemplate` property when no matches found on search.

### CSHTML

```

<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").Fields(field
=>field.Value("id").ParentValue("pid").HasChildren("hasChild").Text("name").
DataSource(ViewBag.dataSource)).Render()
</div>

```

### NORECORDSTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields templateData = new DropDownTreeFields();
 public ActionResult Index()
 {
 DropDownTreeTemplate dropdownTreeTemplate = new
 DropDownTreeTemplate();
 templateData.DataSource = dropdownTreeTemplate.Template();
 templateData.Value = "id";
 templateData.Text = "name";
 templateData.Expanded = "expanded";
 }
 }
}

```

```

 templateData.HasChildren = "hasChild";
 templateData.ParentValue = "pid";
 ViewBag.templateData = templateData;
 ViewBag.localData = templateData.DataSource;
 return View();
 }
}

```

### DROPDOWNTREETEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 return localData;
 }
 }
}

```

Select an employee ▲

NO DATA AVAILABLE

### Action failure template

The Dropdown Tree provides an option to custom design the popup list content using `actionFailureTemplate` property, when the data fetch request fails at the remote server.

### CSHTML

```

<div class="control-section">
 <div style="margin: 0 auto; width:350px;">
 @Html.EJS().DropDownTree("ddtremote").Placeholder("Select a
name").PopupHeight("200px").ActionFailureTemplate("<span class='action-
failure'> Data fetch request
fails").Fields(ViewBag.remoteFields).Render()
 </div>
</div>

```

### ACTIONFAILURETEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

```

```

using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
using Syncfusion.EJ2;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields parentData = new DropDownTreeFields();
 DropDownTreeFields childData = new DropDownTreeFields();
 public ActionResult Index()
 {
 object data = new Syncfusion.EJ2.DataManager
 {
 Url =
 "https://services.odata.org/V4/Northwind/Northwind.svs",
 Adaptor = "ODataV4Adaptor",
 CrossDomain = true
 };
 parentData.Query = "new
ej.data.Query().from('Employees').select('EmployeeID,FirstName,Title').take(
5)";
 parentData.Value = "EmployeeID";
 parentData.Text = "FirstName";
 parentData.HasChildren = "EmployeeID";
 parentData.Child = childData;
 parentData.DataSource = data;
 childData.Query = "new
ej.data.Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5)
";
 childData.Value = "OrderID";
 childData.Text = "ShipName";
 childData.ParentValue = "EmployeeID";
 childData.DataSource = data;
 ViewBag.remoteFields = parentData;
 DropDownTreeFields childDatas = new DropDownTreeFields();
 childDatas.Query = "new
ej.data.Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5)
";
 childDatas.Value = "OrderID";
 childDatas.Text = "ShipName";
 childDatas.ParentValue = "EmployeeID";
 childDatas.DataSource = new DataManager
 {
 Url =
 "https://services.odata.org/V4/Northwind/Northwind.svc",
 Adaptor = "ODataV4Adaptor",
 CrossDomain = true
 };
 ViewBag.child = childDatas;
 return View();
 }
 }
}

```

**DROPDOWNTREETEMPLATE.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 localData.Add(new { id = 1, name = "Steven Buchanan", hasChild =
true, expanded = true, job = "General Manager" });
 localData.Add(new { id = 2, pid = 1, name = "Laura Callahan",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 3, pid = 2, name = "Andrew Fuller",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 4, pid = 3, name = "Anne Dodsworth",
job = "Developer" });
 localData.Add(new { id = 5, pid = 3, name = "Lilly", job =
"Developer", status = "online" });
 localData.Add(new { id = 6, pid = 1, name = "Nancy Davolio",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 7, pid = 6, name = "Michael Suyama",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 8, pid = 7, name = "Robert King", job
= "Developer" });
 localData.Add(new { id = 9, pid = 7, name = "Mary", job =
"Developer" });
 localData.Add(new { id = 10, pid = 1, name = "Janet Leverling",
job = "HR" });
 return localData;
 }
 }
}

```

Select an employee

Data fetch request fails

### Custom template to show selected items in input

In Dropdown Tree, while selecting more than one item via checkbox or multi selection support, all the selected items will be displayed in the input. Instead of displaying all the selected item text, the custom template can be displayed by setting the **mode** property as **Custom** and **customTemplate** property.

When the **mode** property is set as **Custom**, the Dropdown Tree displays the default template value (**\${value.length} item(s) selected**) like **1 item(s) selected** or **2 item(s) selected**. The default template can be customized by setting **customTemplate** property.

In the following sample, the Dropdown Tree is rendered with default value of the **customTemplate** property like **"1 item(s) selected or 2 item(s) selected"**.

### CSHTML

```
@using Syncfusion.EJ2.DropDowns
```

```
<div class="control-section">
 <div style="margin: 0 auto; width:350px;">
 @Html.EJS().DropDownTree("ddt").Placeholder("Select an
employee").Mode(Mode.Custom).CustomTemplate("${value.length} item(s)
selected").ShowCheckBox(true).PopupHeight("200px").Fields(ViewBag.templateDa
ta).Render()
 </div>
</div>
```

### CUSTOMTEMPLATEMODE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields templateData = new DropDownTreeFields();
 public ActionResult Index()
 {
 DropDownTreeTemplate dropdownTreeTemplate = new
DropDownTreeTemplate();
 templateData.DataSource = dropdownTreeTemplate.Template();
 templateData.Value = "id";
 templateData.Text = "name";
 templateData.Expanded = "expanded";
 templateData.HasChildren = "hasChild";
 templateData.ParentValue = "pid";
 ViewBag.templateData = templateData;
 //ASP.NET Core Code Blocks
 ViewBag.localData = templateData.DataSource;
 return View();
 }
 }
}
```

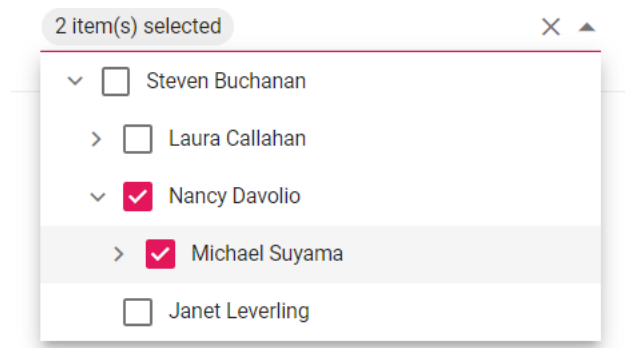
### DROPDOWNTREETEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 localData.Add(new { id = 1, name = "Steven Buchanan", hasChild =
true, expanded = true, job = "General Manager" });
 }
 }
}
```

```

 localData.Add(new { id = 2, pid = 1, name = "Laura Callahan",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 3, pid = 2, name = "Andrew Fuller",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 4, pid = 3, name = "Anne Dodsworth",
job = "Developer" });
 localData.Add(new { id = 5, pid = 3, name = "Lilly", job =
"Developer", status = "online" });
 localData.Add(new { id = 6, pid = 1, name = "Nancy Davolio",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 7, pid = 6, name = "Michael Suyama",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 8, pid = 7, name = "Robert King", job
= "Developer" });
 localData.Add(new { id = 9, pid = 7, name = "Mary", job =
"Developer" });
 localData.Add(new { id = 10, pid = 1, name = "Janet Leverling",
job = "HR" });
 return localData;
 }
}

```



In the following sample, the DropDown Tree is rendered with custom value of the **customTemplate** property like **Selected items count: 2**.

### CSHTML

```

@using Syncfusion.EJ2.DropDowns
<div class="control-section">
 <div style="margin: 0 auto; width:350px;">
 @Html.EJS().DropDownTree("ddt").Placeholder("Select an
employee").Mode(Mode.Custom).CustomTemplate("Selected item(s) count:
${value.length}").ShowCheckBox(true).PopupHeight("200px").Fields(ViewBag.tem
plateData).Render()
 </div>
</div>

```

### CUSTOMTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

```



```

using System.Web.Mvc;
using itemtemplates.Models;
using Syncfusion.EJ2.DropDowns;
namespace itemtemplates.Controllers
{
 public class HomeController : Controller
 {
 DropDownTreeFields templateData = new DropDownTreeFields();
 public ActionResult Index()
 {
 //ASP.NET MVC Code Blocks
 DropDownTreeTemplate dropdownTreeTemplate = new
DropDownTreeTemplate();
 templateData.DataSource = dropdownTreeTemplate.Template();
 templateData.Value = "id";
 templateData.Text = "name";
 templateData.Expanded = "expanded";
 templateData.HasChildren = "hasChild";
 templateData.ParentValue = "pid";
 ViewBag.templateData = templateData;
 //ASP.NET Core Code Blocks
 ViewBag.localData = templateData.DataSource;
 return View();
 }
 }
}

```

### DROPDOWNTREETEMPLATE.CS

```

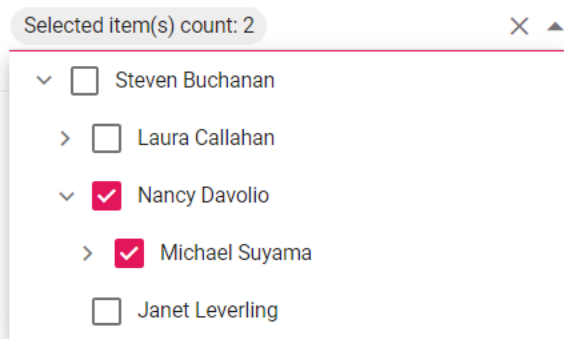
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace itemtemplates.Models
{
 public class DropDownTreeTemplate
 {
 public List<Object> Template()
 {
 List<object> localData = new List<object>();
 localData.Add(new { id = 1, name = "Steven Buchanan", hasChild =
true, expanded = true, job = "General Manager" });
 localData.Add(new { id = 2, pid = 1, name = "Laura Callahan",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 3, pid = 2, name = "Andrew Fuller",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 4, pid = 3, name = "Anne Dodsworth",
job = "Developer" });
 localData.Add(new { id = 5, pid = 3, name = "Lilly", job =
"Developer", status = "online" });
 localData.Add(new { id = 6, pid = 1, name = "Nancy Davolio",
job = "Product Manager", hasChild = true });
 localData.Add(new { id = 7, pid = 6, name = "Michael Suyama",
job = "Team Lead", hasChild = true });
 localData.Add(new { id = 8, pid = 7, name = "Robert King", job
= "Developer" });
 }
 }
}

```

```

 localData.Add(new { id = 9, pid = 7, name = "Mary", job =
"Developer" });
 localData.Add(new { id = 10, pid = 1, name = "Janet Leverling",
job = "HR" });
 return localData;
 }
}

```



## CheckBox

The Dropdown Tree control allows to check more than one item from the tree without affecting the UI's appearance by enabling the `showCheckBox` property. When this property is enabled, checkbox appears before each item text in the popup.

In the following example, the `showCheckBox` property is enabled.

### CSHTML

```

<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").ShowCheckBox(true).Fields(field =>
 field.Value("id").ParentValue("pid").HasChildren("hasChild").Text("name").Da
 taSource(ViewBag.dataSource)).Render()
</div>

```

### CHECKBOX.CS

```

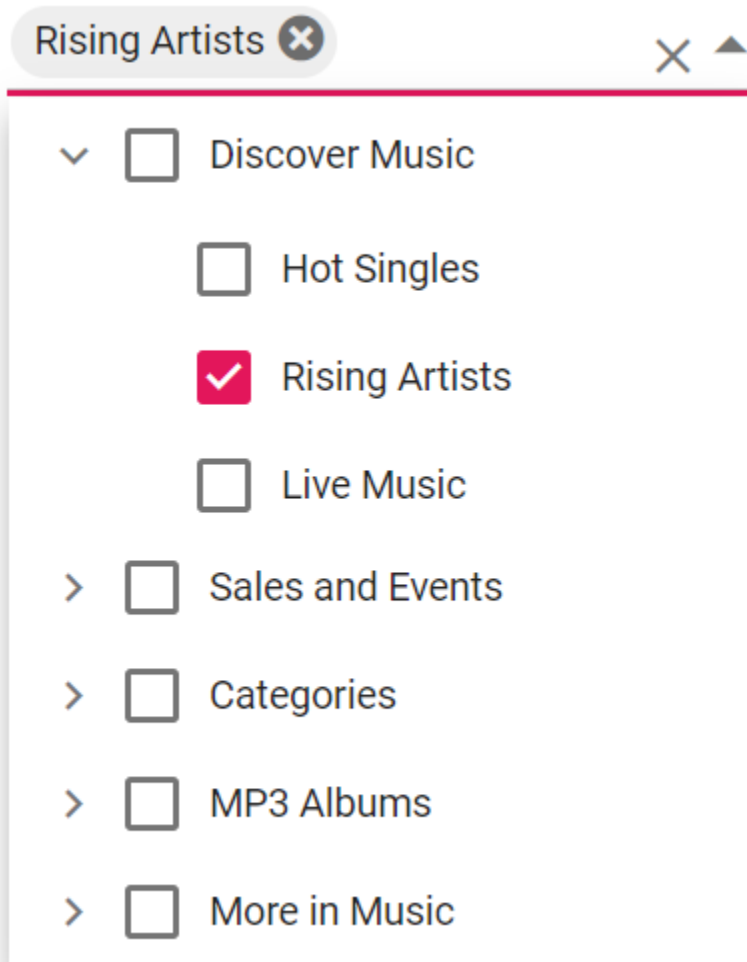
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownList.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownList.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult CheckBox()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new

```

```
{
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true
});
treedata.Add(new
{
 id = 2,
 pid = 1,
 name = "Hot Singles",
});
treedata.Add(new
{
 id = 3,
 pid = 1,
 name = "Rising Artists"
});
treedata.Add(new
{
 id = 4,
 pid = 1,
 name = "Live Music"
});
treedata.Add(new
{
 id = 5,
 hasChild = true,
 name = "Sales and Events",
});
treedata.Add(new
{
 id = 6,
 pid = 5,
 name = "100 Albums - $5 Each",
});
treedata.Add(new
{
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
});
treedata.Add(new
{
 id = 8,
 pid = 5,
 name = "CD Deals"
});
treedata.Add(new
{
 id = 10,
 hasChild = true,
 name = "Categories"
});
treedata.Add(new
{
 id = 11,
```

```
 pid = 10,
 name = "Bestselling Albums",
 });
 treedata.Add(new
 {
 id = 12,
 pid = 10,
 name = "New Releases"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
 });
 treedata.Add(new
 {
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 14,
 name = "Rock"
 });
 treedata.Add(new
 {
 id = 16,
 name = "Gospel",
 pid = 14,
 });
 treedata.Add(new
 {
 id = 17,
 pid = 14,
 name = "Latin Music"
 });
 treedata.Add(new
 {
 id = 18,
 pid = 14,
 name = "Jazz"
 });
 treedata.Add(new
 {
 id = 19,
 hasChild = true,
 name = "More in Music"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 19,
 name = "Music Trade-In"
 });
});
```

```
 treedata.Add(new
 {
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
 });
 treedata.Add(new
 {
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
 });
 ViewBag.dataSource = treedata;
 return View();
 }
}
```



### Auto Check

By default, the checkbox state of the parent and child items in the Dropdown Tree will not be dependent over each other. If you need dependent checked state, then enable the `autoCheck` property which is a member of `treeSettings` property.

- If one or more child items are not in the checked state, then the parent item will be in the intermediate state.
- If all the child items are checked, then the parent item will also be in the checked state.
- If a parent item is checked, then all the child items will also be changed to checked state.

In the following example, the `autoCheck` property is enabled.

### CSHTML

```
<div id='container' style="margin:0 auto; width:250px;">
 @Html.EJS().DropDownTree("treedata").ShowCheckBox(true).Fields(field =>
 field.Value("id").ParentValue("pid").HasChildren("hasChild").Text("name").DataSource(ViewBag.dataSource)).TreeSettings(treeSettings =>
 treeSettings.AutoCheck(true)).Render()
</div>
```

### AUTOCHECK.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownTree.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownTree.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult AutoCheck()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
```

```
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Live Music"
 });
 treedata.Add(new
 {
 id = 5,
 hasChild = true,
 name = "Sales and Events",
 });
 treedata.Add(new
 {
 id = 6,
 pid = 5,
 name = "100 Albums - $5 Each",
 });
 treedata.Add(new
 {
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
 });
 treedata.Add(new
 {
 id = 8,
 pid = 5,
 name = "CD Deals"
 });
 treedata.Add(new
 {
 id = 10,
 hasChild = true,
 name = "Categories"
 });
 treedata.Add(new
 {
 id = 11,
 pid = 10,
 name = "Bestselling Albums",
 });
 treedata.Add(new
 {
 id = 12,
 pid = 10,
 name = "New Releases"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
```

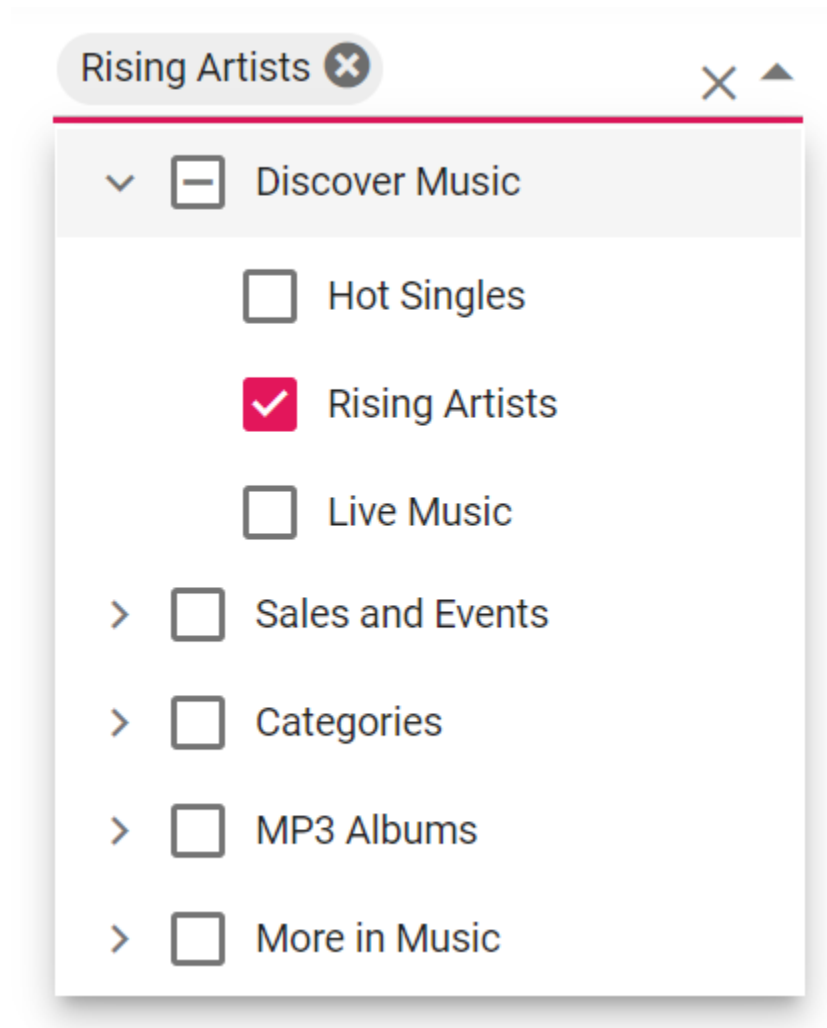
```
});
treedata.Add(new
{
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
});
treedata.Add(new
{
 id = 15,
 pid = 14,
 name = "Rock"
});
treedata.Add(new
{
 id = 16,
 name = "Gospel",
 pid = 14,
});
treedata.Add(new
{
 id = 17,
 pid = 14,
 name = "Latin Music"
});
treedata.Add(new
{
 id = 18,
 pid = 14,
 name = "Jazz"
});
treedata.Add(new
{
 id = 19,
 hasChild = true,
 name = "More in Music"
});
treedata.Add(new
{
 id = 20,
 pid = 19,
 name = "Music Trade-In"
});
treedata.Add(new
{
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
});
treedata.Add(new
{
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
});
ViewBag.dataSource = treedata;
return View();
```



```

 }
}
}

```



### Select All

The Dropdown Tree control has in-built support to select all the tree items using Select All options in the header.

When the `showSelectAll` property is set to true, a checkbox will be displayed in the popup header that allows to select or deselect all the tree items in the popup.

By default, `Select All` and `unSelect All` text values will be showcased along with the checkbox in the popup header to indicate the action to be performed on checking or unchecking the checkbox. You can customize these name attributes by using `selectAllText` and `unSelectAllText` properties respectively.

### CSHTML

```

<div id='container' style="margin:0 auto; width:250px;">

@Html.EJS().DropDownTree("treedata").ShowCheckBox(true).ShowSelectAll(true).
SelectAllText("Check All").UnSelectAllText("UnCheck All").Fields(field =>

```

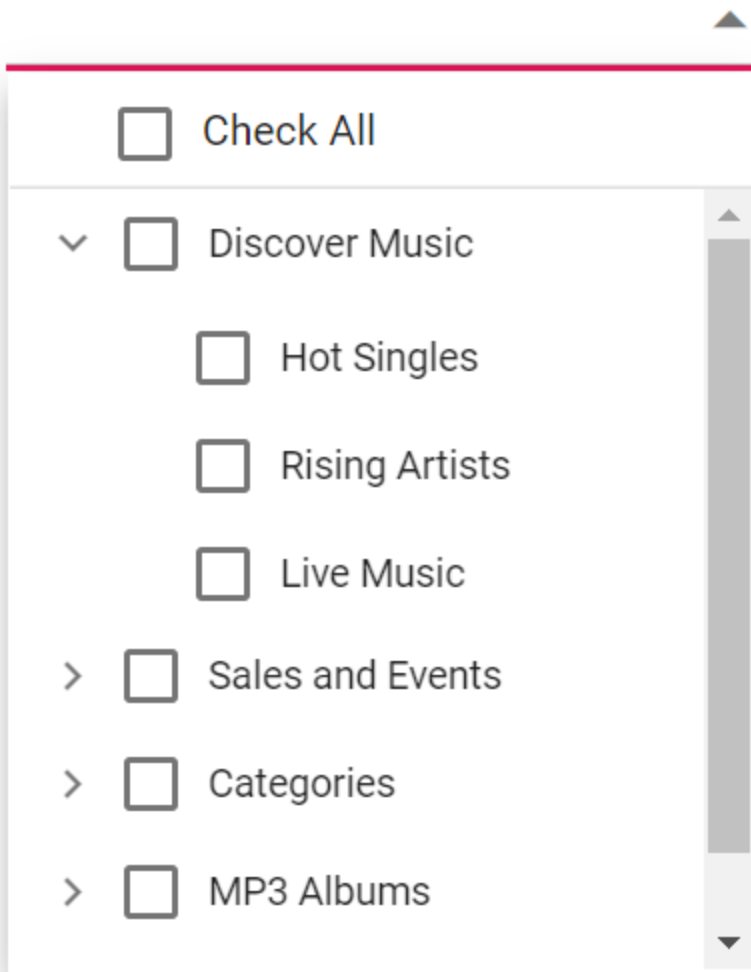
```
field.Value("id").ParentValue("pid").HasChildren("hasChild").Text("name").DataSource(ViewBag.dataSource)).Render()
</div>
```

### SELECTALL.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using DropDownTree.Models;
using Syncfusion.EJ2.DropDowns;
namespace DropDownTree.Controllers
{
 public class DropDownListController : Controller
 {
 public IActionResult SelectAll()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Live Music"
 });
 treedata.Add(new
 {
 id = 5,
 hasChild = true,
 name = "Sales and Events",
 });
 treedata.Add(new
 {
 id = 6,
```

```
 pid = 5,
 name = "100 Albums - $5 Each",
 });
 treedata.Add(new
 {
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
 });
 treedata.Add(new
 {
 id = 8,
 pid = 5,
 name = "CD Deals"
 });
 treedata.Add(new
 {
 id = 10,
 hasChild = true,
 name = "Categories"
 });
 treedata.Add(new
 {
 id = 11,
 pid = 10,
 name = "Bestselling Albums",
 });
 treedata.Add(new
 {
 id = 12,
 pid = 10,
 name = "New Releases"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
 });
 treedata.Add(new
 {
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 14,
 name = "Rock"
 });
 treedata.Add(new
 {
 id = 16,
 name = "Gospel",
 pid = 14,
 });
 });
```

```
 treedata.Add(new
 {
 id = 17,
 pid = 14,
 name = "Latin Music"
 });
 treedata.Add(new
 {
 id = 18,
 pid = 14,
 name = "Jazz"
 });
 treedata.Add(new
 {
 id = 19,
 hasChild = true,
 name = "More in Music"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 19,
 name = "Music Trade-In"
 });
 treedata.Add(new
 {
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
 });
 treedata.Add(new
 {
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
 });
 ViewBag.dataSource = treedata;
 return View();
 }
}
```



### Accessibility in ASP.NET MVC Dropdown Tree component

The Dropdown Tree component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dropdown Tree component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The Dropdown Tree component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Dropdown Tree component:

| Attributes | Purpose |

| --- | --- |

| **role=listbox** | All list items are contained within the element. |

| **aria-disabled** | Indicates element is perceivable but disabled. |

| **aria-owns** | This attribute contains the ID of the popup list to indicate popup as a child element. |

| **aria-haspopup** | Indicates whether the Dropdown Tree input element has a popup list or not. |

| **aria-expanded** | Indicates the state of the popup list for Dropdown Tree and the parent node's expansion status for TreeView. |

| **aria-activedescendent** | This attribute holds the ID of the active list item to focus its descendant child element. |

| **aria-labelledby** | This attribute points to the element(s) labeling the element it's applied to. |

- | **aria-decribedby** | This attribute points to the element(s) describing the one it's set on. |
- | **role=tree** | All tree nodes are contained within the element. |
- | **role=treeitem** | Specifies the role of each tree node in a selectable TreeView and its containment within the tree. |
- | **role=group** | Specifies the role of each parent node container in the TreeView. |
- | **role=checkbox** | Indicates checkbox control along with treeitem element. |
- | **aria-multiselectable** | Indicates whether the TreeView enables multiple selection or not. |
- | **aria-selected** | Indicates the selected node. |
- | **aria-level** | Indicates the level of node in TreeView. |
- | **aria-checked** | Indicates the current checked state of TreeView checkbox. |
- | **aria-label** | Indicates the contextual message for the TreeView checkbox and Dropdown Tree. |
- | **aria-activedescendant** | Identifies the currently active element when focusing on the TreeView. |

#### Keyboard interaction

The Dropdown Tree component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Dropdown Tree component.

| Interaction Keys | Description |

|-----|-----|

| **Alt + Down** | Opens the popup. |

| **Alt + Up** | Closes the popup. |

| **Esc(Escape)** | Closes the popup when it is in an open state. |

| **Arrow Up** | Goes to the previous item in the popup. |

| **Arrow Down** | Goes to the next item in the popup. |

| **Arrow Right** | Expands the current item in the popup. |

| **Arrow Left** | Collapses the current item in the popup. |

| **Home** | Goes to the first item in the popup. |

| **End** | Goes to the last item in the popup. |

| **Enter** | Selects the focused item in the popup. |

| **Space** | Checks the current item in the popup. |

#### Ensuring accessibility

The Dropdown Tree component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dropdown Tree component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dropdown Tree component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## Localization in Dropdown Tree Component

The Dropdown Tree component can be localized to any culture by defining the texts and messages of the Dropdown Tree in the corresponding culture. The default locale of the Dropdown Tree is **en** (English). The following table represents the default texts and messages of the Dropdown Tree in **en** culture.

KEY Text/Message
---- ----
noRecordsTemplate No records found
actionFailureTemplate Request failed
overflowCountTemplate +\${count} more..
totalCountTemplate \${count} selected

## File Manager

### Getting Started with ASP.NET MVC FileManager Control

This section briefly explains about how to include [ASP.NET MVC FileManager](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.



**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

##### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

##### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

#### Add ASP.NET MVC FileManager Control

Now, add the Syncfusion ASP.NET MVC FileManager control in **~/Views/Home/Index.cshtml** page.

##### CSHTML

```
<div class="control-section">
 <div class="sample-container">
```

```

 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations"
 }).NavigationPaneSettings(ls =>
ls.Visible(false)).ContextMenuSettings(ls =>
ls.Visible(false)).ToolbarSettings(ls => ls.Visible(false)).Render()
 <!-- end of filemanager element -->
 </div>
</div>

```

## HOMECONTROLLER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":

```

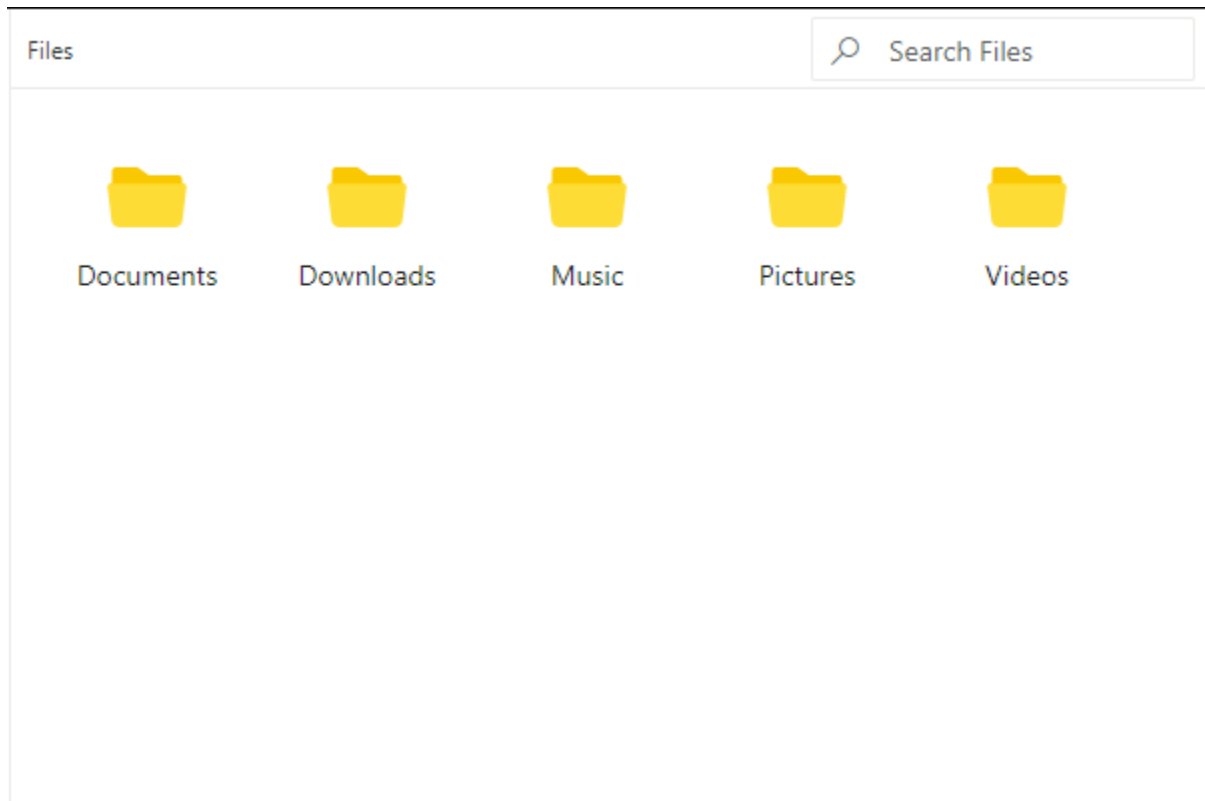
```

 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
 }
 // Processing the Upload operation
 public ActionResult Upload(string path,
 IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
 {
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
 }

```

```
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC FileManager control will be rendered in the default web browser.



**Note:** The File Manager can be rendered with **local service** for sending ajax request. Ajax request will be sent to the server which then processes the request and sends back the response. Refer Controller file for file manager service.

### File Download support

To perform the download operation, initialize the **DownloadUrl** property in a [AjaxSettings](#) of File Manager component.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 DownloadUrl = "/Home/Download"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### File Upload support

To perform the upload operation, initialize the **UploadUrl** property in a [AjaxSettings](#) of File Manager Component.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 UploadUrl = "/Home/Upload"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

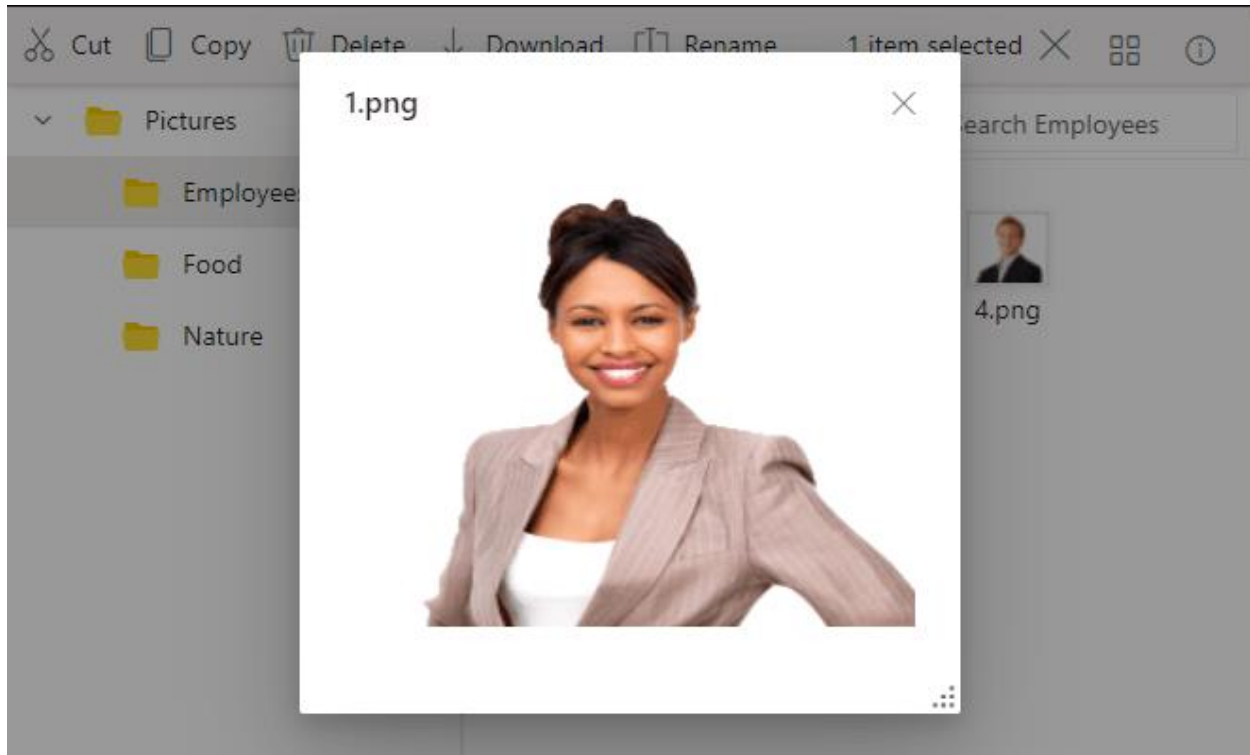
### Image Preview support

To perform the image preview support in the File Manager component, need to initialize the [GetImageUrl](#) property in a [AjaxSettings](#) of File Manager component.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
```

```
 GetImageUrl = "/Home/GetImage"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```



### File Manager Overview

By default, the File Manager component having extra module like [NavigationPane](#), [Toolbar](#), [ContextMenu](#) module.

In this sample demonstrates the full features of the File Manager that includes toolbar, navigation pane and details view.

### HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;

// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 }
}
```

```

PhysicalFileProvider operation = new PhysicalFileProvider();
public HomeController()
{
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
}
public ActionResult FileOperations(FileManagerDirectoryContent args)
{
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 }
}

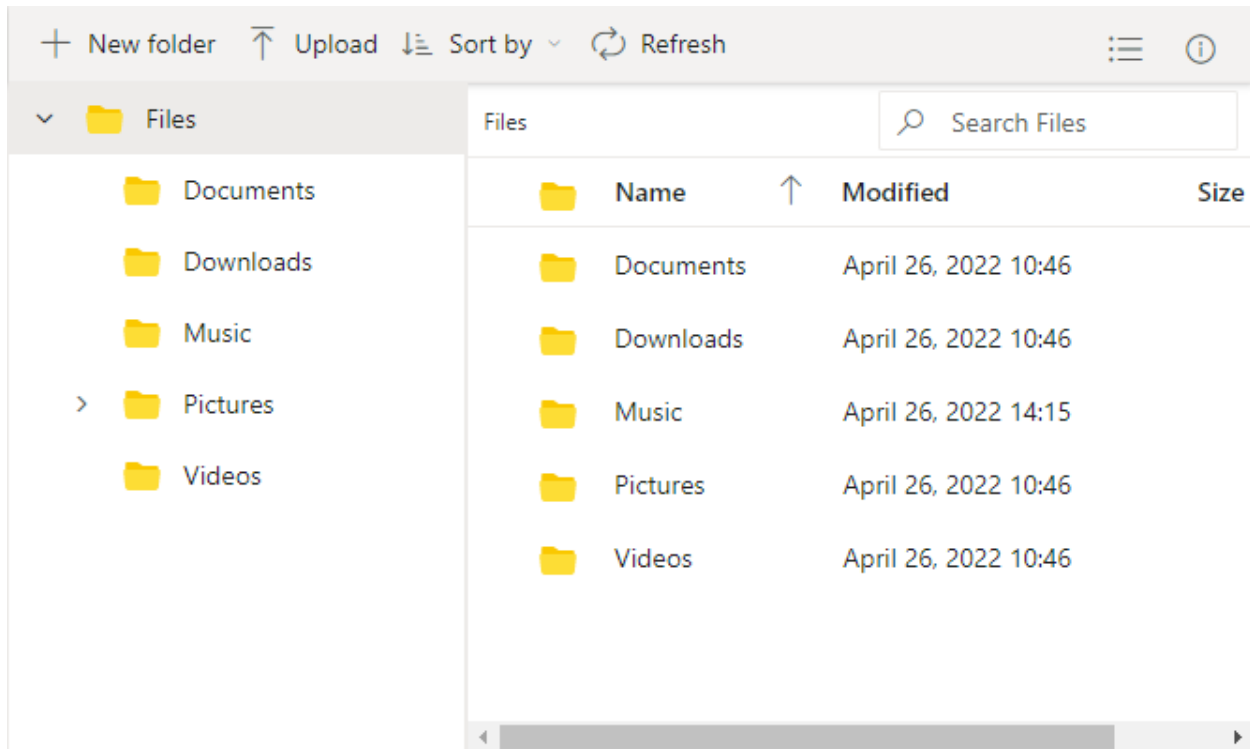
```

```

 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```





**Note:** The appearance of the File Manager can be customized by using [cssClass](#) property. This adds a css class to the root of the File Manager which can be used to add new styles or override existing styles to the File Manager.

#### Switching initial view of the File Manager

The initial view of the File Manager can be changed to details or largeicons view with the help of [View](#) property. By default, the File Manager will be rendered in large icons view. When the File Manager is initially rendered, [Created](#) will be triggered. This event can be utilized for performing operations once the File Manager has been successfully created.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).View(Syncfusion.EJ2.FileManager.ViewType.Details).Created("onCreated").Render()
 <!-- end of filemanager element -->
 </div>
</div>
<script>
 // File Manager's created event function
 function onCreated() {
```

```
 console.log("File Manager has been created successfully");
 }
</script>
```

## HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
```

```

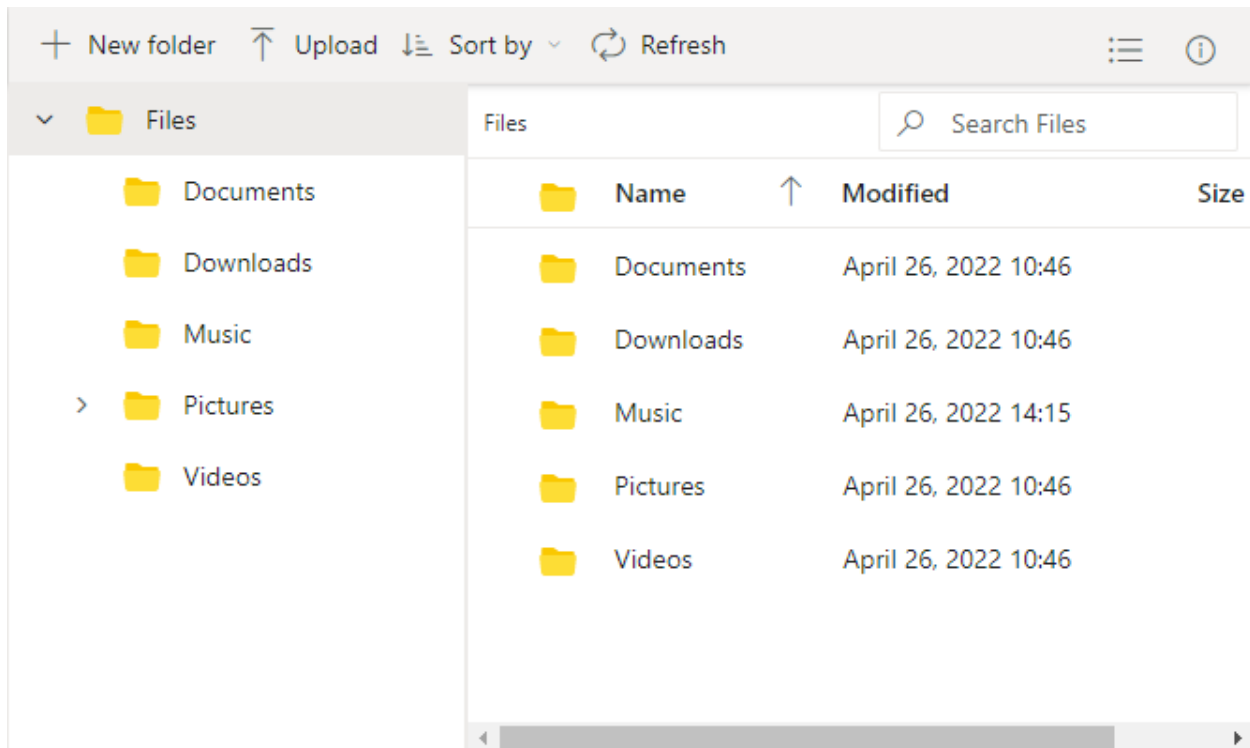
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 }
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;

```

```

 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
 }
 public ActionResult Index()
 {
 return View();
 }
}
}

```



#### Maintaining component state on page reload

The File Manager supports maintaining the component state on page reload. This can be achieved by enabling [EnablePersistence](#) property which maintains the following,

- Previous view of the File Manager - [View](#)
- Previous path of the File Manager - [Path](#)
- Previous selected items of the File Manager - [SelectedItems](#)

For every operation in File Manager, ajax request will be sent to the server which then processes the request and sends back the response. When the ajax request is success, [Success](#) event will be triggered and [Failure](#) event will be triggered if the request gets failed.

**CSHTML**

```

<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).EnablePersistence(true).Success("onSuccess").Failure("onFailure").Render(
)
 <!-- end of filemanager element -->
 </div>
</div>
<script>
 // File Manager's file onSuccess function
 function onSuccess() {
 console.log("Ajax request successful");
 }
 // File Manager's file onError function
 function onFailure() {
 console.log("Ajax request has failed");
 }
</script>

```

**HOMECONTROLLER.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 }
}

```

```

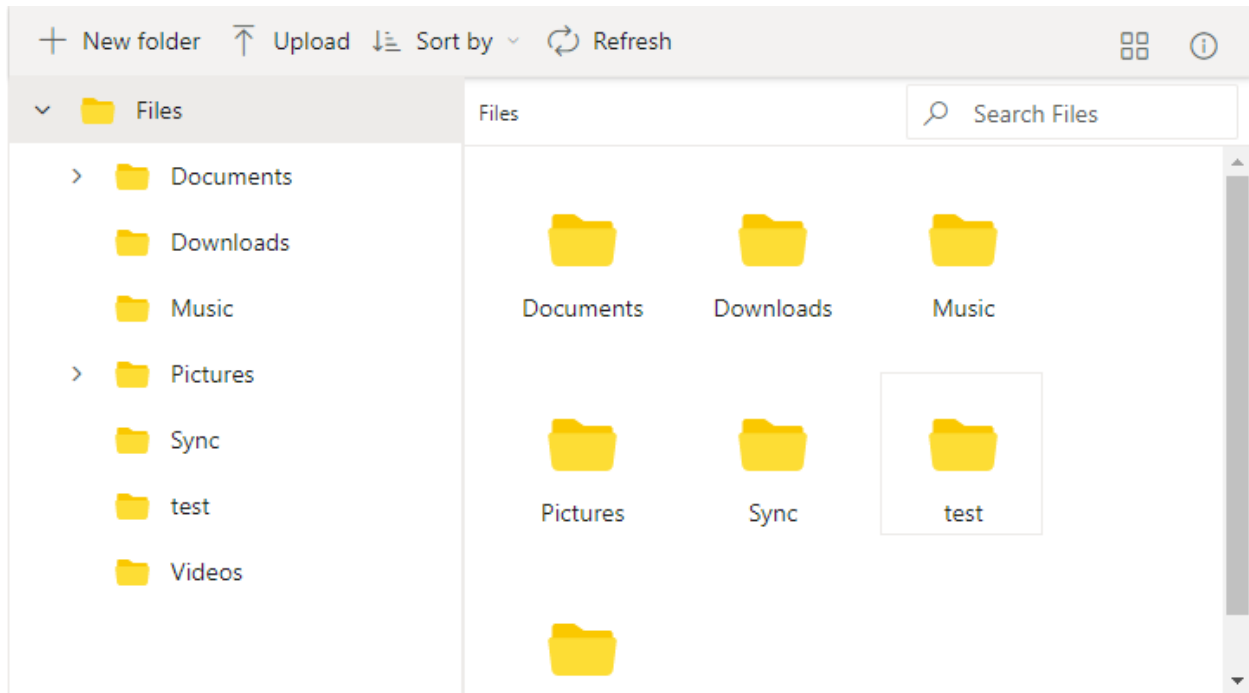
{
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 // SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 // which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 // file name; NewName - New file name
 }
}

```

```

 return
Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
}

```



**Note:** The files of the current folder opened in the File Manager can be refreshed programmatically by calling `refreshFiles` method

Rendering component in right-to-left direction

It is possible to render the File Manager in right-to-left direction by setting the [EnableRtl](#) API to true.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).EnableRtl(true).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
```



```

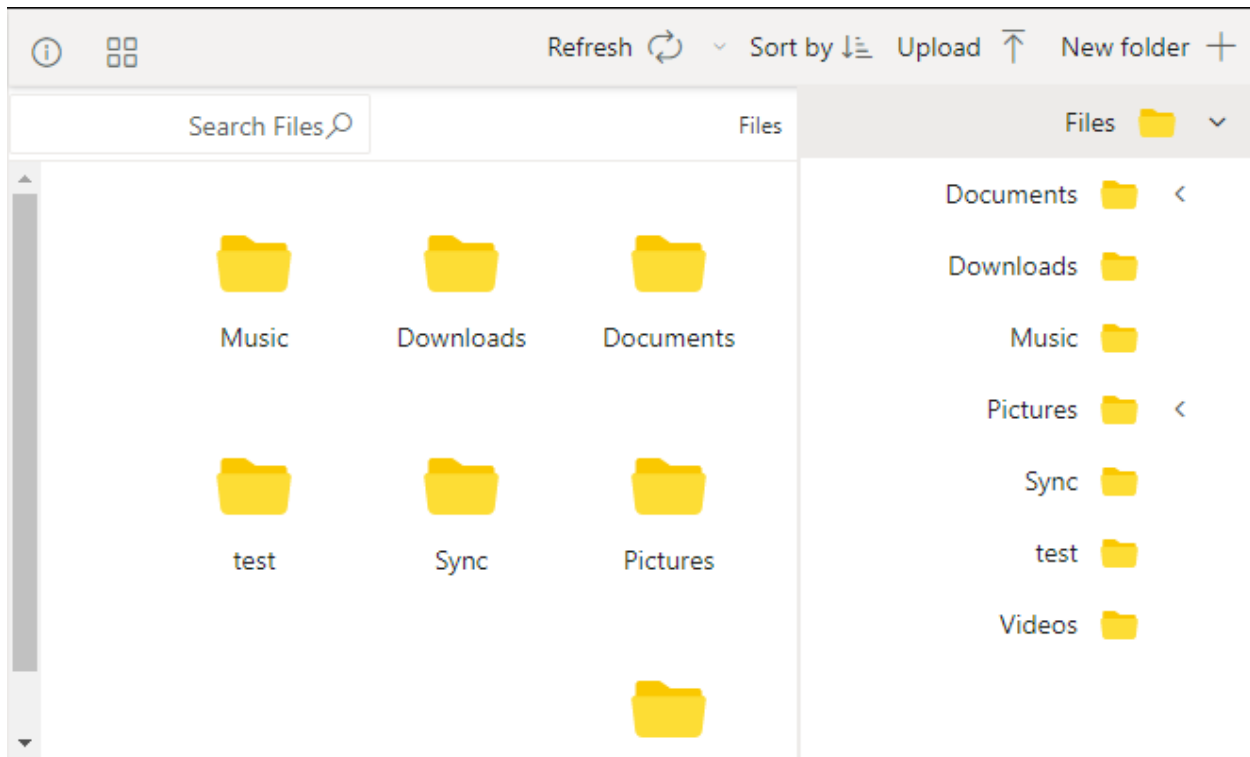
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 }
 }
 }
}

```

```

 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 // SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 // which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 // file name; NewName - New file name
 return
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
 IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to be uploaded;
 // uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 // Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```



### Specifying the current path of the File Manager

The current path of the File Manager can be specified initially or dynamically using the [Path](#) property.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).Path("/Pictures/Employees").Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

#### HOMECONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
```

```

using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders

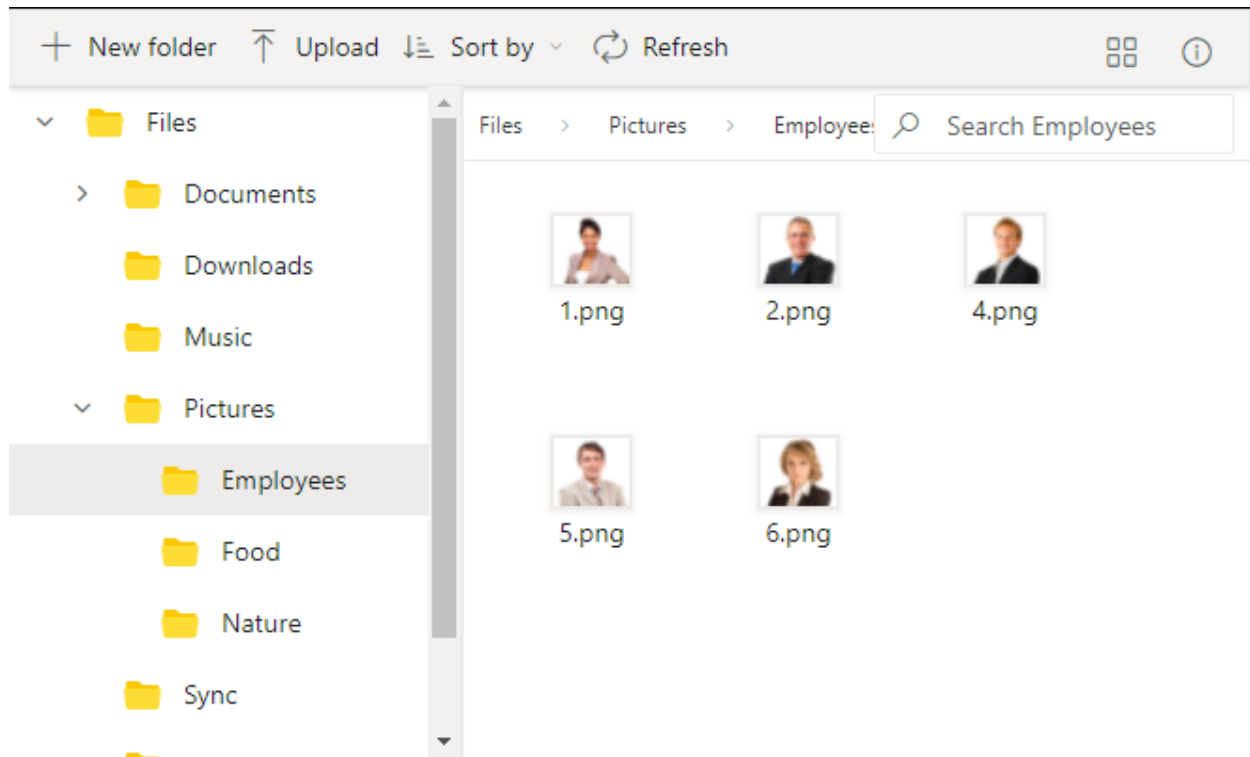
```

```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 // SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 // which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 // file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 // uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 // Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}

```

```
}
}
```

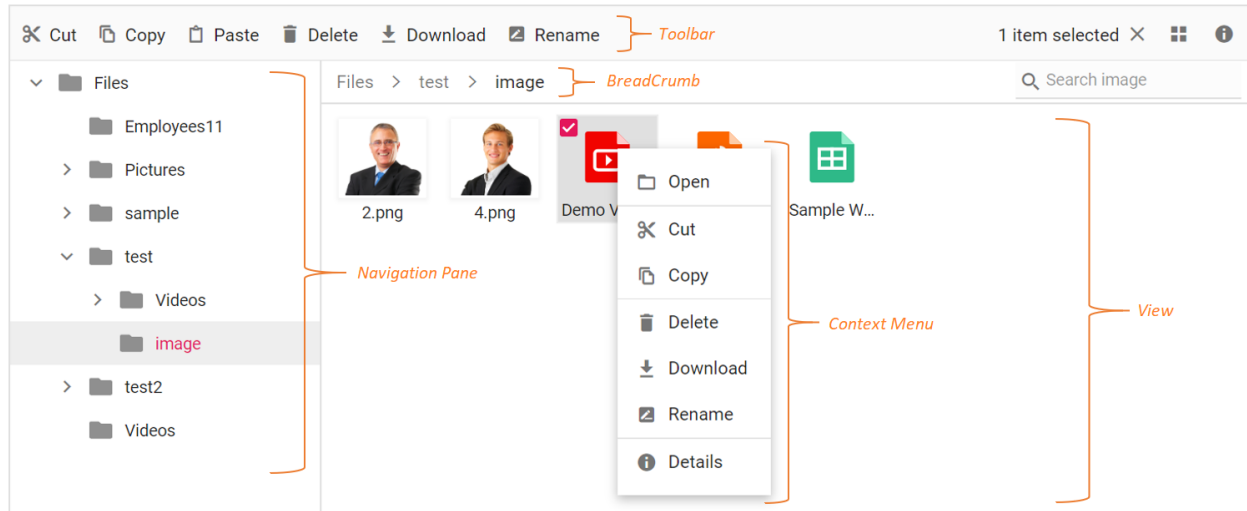


**Note:** [View Sample in GitHub.](#)

### User Interface Structure

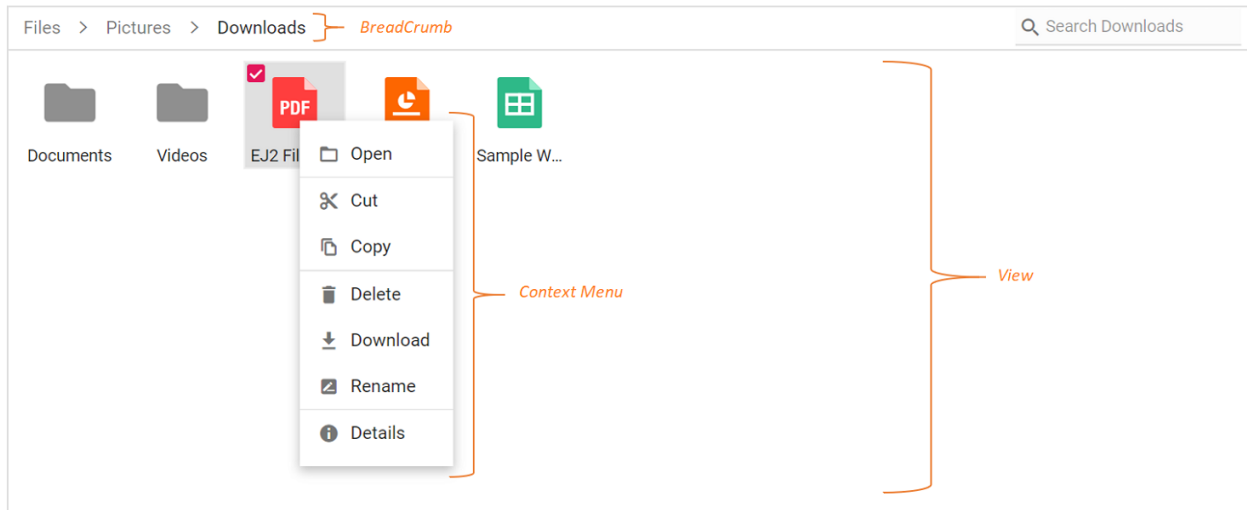
The file manager UI is comprised of several sections like view, toolbar, breadcrumb, context menu, and so on. The UI of the file manager is enhanced with injectable modules like **Details View** for browsing files and folders in a grid, **Navigation Pane** for folder navigation, and **Toolbar** for file operations. The file manager with all feature modules have the following sections in its UI.

- [Toolbar](#) (For direct access to file operations)
- [Navigation Pane](#) (For easy navigation between folders)
- [Breadcrumb](#) (For parent folder navigations)
- [View](#) (For browsing files and folders using large icon view or details view)
- [Context Menu](#) (For accessing file operations)



The basic file manager is a light weight component with all the basic functions. The basic file manager have the following sections in its UI to browse files and folders and manage them with file operations.

- [Breadcrumb](#) (For parent folder navigations)
- [View](#) (Large Icons view for browsing files and folders)
- [Context Menu](#) (For accessing file operations)

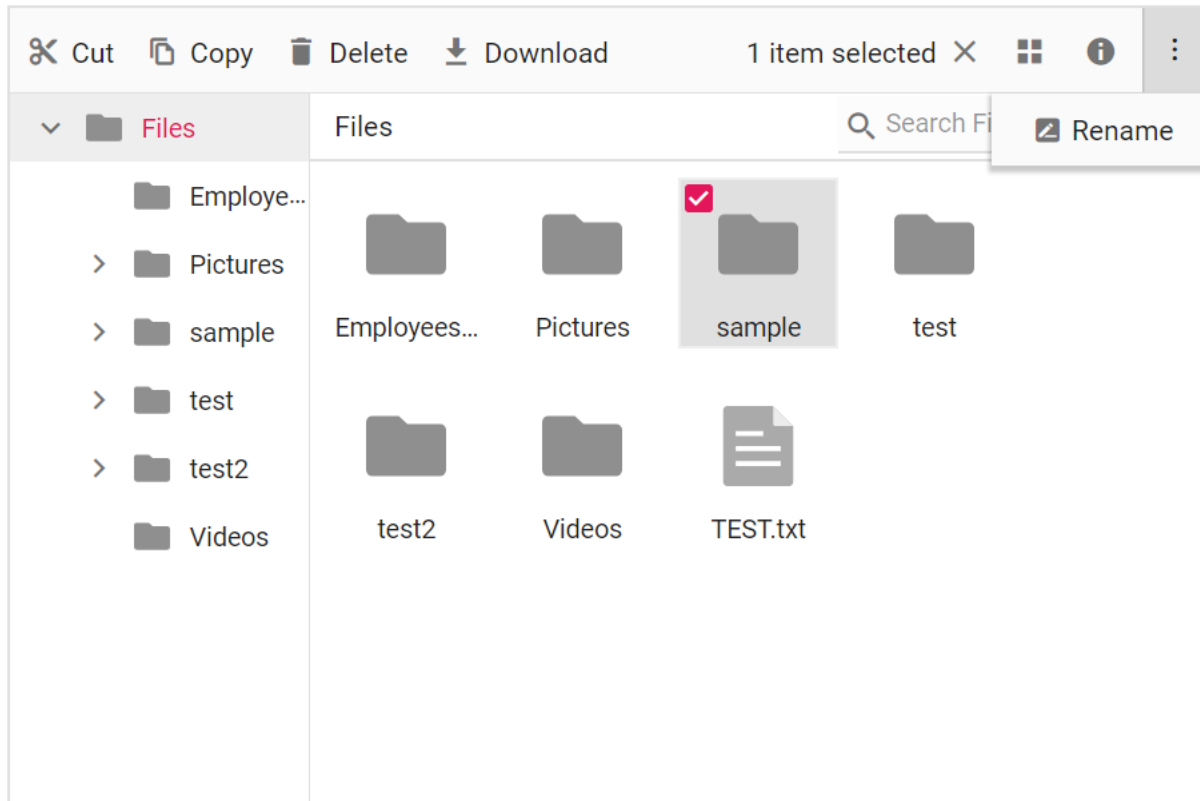


### Toolbar

The **Toolbar** provides easy access to the file operations using different buttons and it is presented at the top of the file manager.

If the toolbar items exceed the size of the toolbar, then the exceeding toolbar size will be moved to toolbar popup with a dropdown button at the end of toolbar.

*\*Refer [Toolbar](#) section in file operations to know more about the buttons present in toolbar\*.*



### Files and folders navigation

The file manager provides navigation between files and folders using the following two options.

- [Navigation Pane](#)
- [Breadcrumb](#)

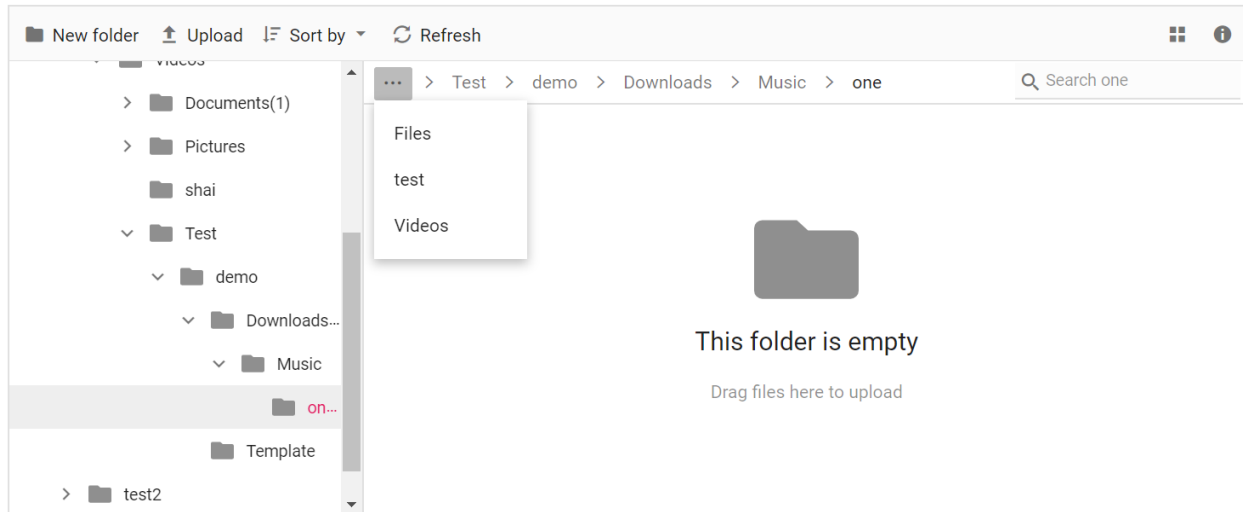
#### Navigation pane

FileManager provides navigation pane, it displays the folder hierarchy of the file system and provides easy navigation to the desired folder. Using `navigationPaneSettings` minimum and maximum width of the navigation pane can be changed. The navigation pane can be shown or hidden using the `visible` option in the `navigationPaneSettings`.

#### BreadCrumb

The file manager provides breadcrumb for navigating to the parent folders. The breadcrumb the in file manager is responsible for resizing. Whenever the path length exceeds the breadcrumb length, a dropdown button will be added at the starting of the breadcrumb to hold the parent folders adjacent to root.





### View

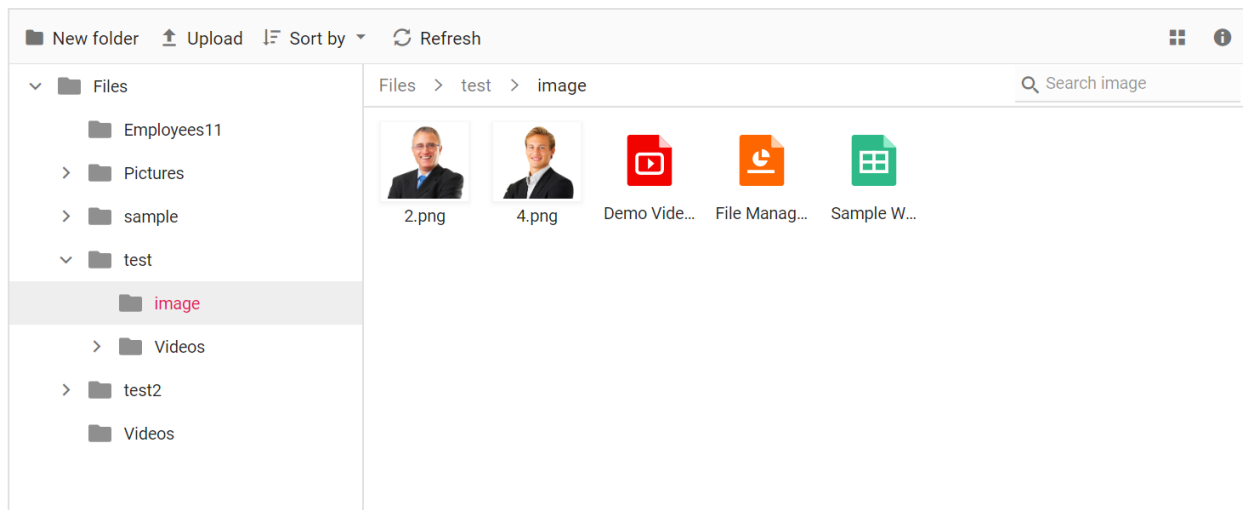
View is the section where the files and folders are displayed for the user to browse. The file manager has two types of views to display the files and folders.

- [Large Icons View](#)
- [Details View](#)

The **large icons view** is the default starting view in the file manager. The view can be changed by using the **toolbar** view button or by using the view menu in **context menu**. The **view** API can also be used to change the initial view of the file manager.

#### Large icons view

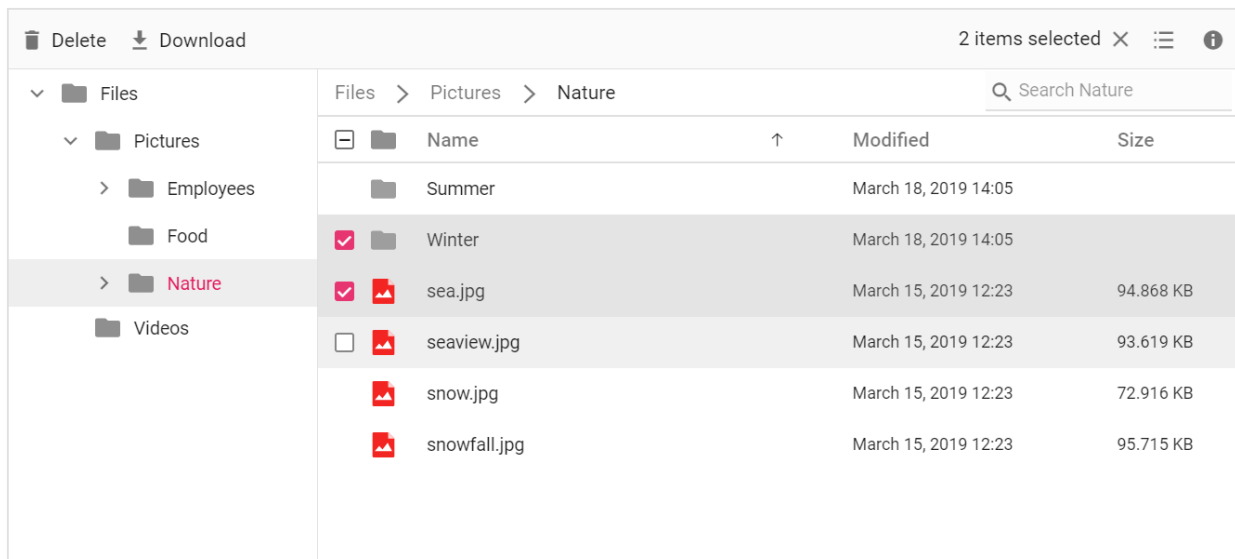
In the large icons view, the thumbnail icons will be shown in a larger size, which displays the data in a form that best suits their content. For image and video type files, a **preview** will be displayed. Extension thumbnails will be displayed for other type files.



#### Details view

FileManager can be changed from large icon to details view by using **View** property. In the details view, the files are displayed in a sorted list order. This file list comprises of several columns of information

about the files such as **Name**, **Date Modified**, **Type**, and **Size**. Each file has its own small icon representing the file type. Additional columns can be added using `detailsViewSettings` API. The details view allows you to perform sorting using column header.

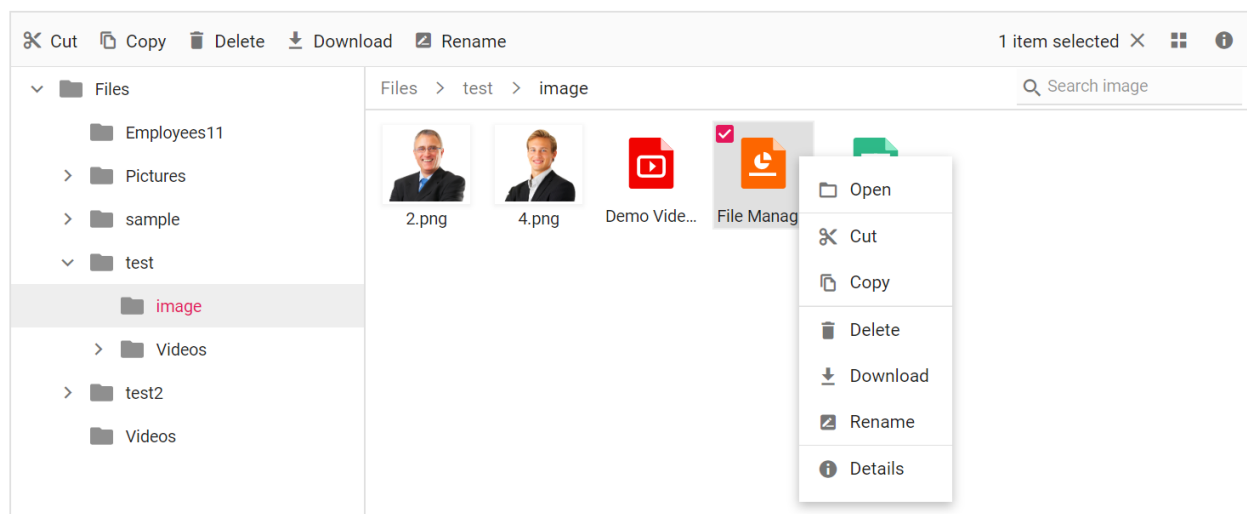


### Context menu

The context menu appears on user interaction such as right-click. The file manager is provided with context menu support to perform list of file operations with the files and folders. Context menu appears with varying menu items based on the targets such as file, folder (including navigation pane folders), and layout (empty area in view).

Context menu can be customized using the `contextMenuSettings`, `menuOpen`, and `menuClick` events.

*\*Refer [Context Menu](#) section in file operations to know more about the menu items present in context menu\*.*



### File Operations in FileManager Component

The file manager component is used to browse, manage, and organize the files and folders in a file system through a web application. All basic file operations like creating a new folder, uploading and downloading of files in the file system, and deleting and renaming of existing files and folders are available in the file manager component. Additionally, previewing of image files is also provided in the file manager component.

The following table represents the basic operations available in the file manager and their corresponding functions.

Operation Name	Function
----------------	----------

----	----
------	------

read	Read the details of files or folders available in the given path from the file system, to display the files for the user to browse the content.
------	-------------------------------------------------------------------------------------------------------------------------------------------------

create	Creates a new folder in the current path of the file system.
--------	--------------------------------------------------------------

delete	Removes the file or folder from the file server.
--------	--------------------------------------------------

rename	Rename the selected file or folder in the file system.
--------	--------------------------------------------------------

search	Searches for items matching the search string in the current and child directories.
--------	-------------------------------------------------------------------------------------

details	Gets the detail of the selected item(s) from the file server.
---------	---------------------------------------------------------------

copy	Copy the selected file or folder in the file system.
------	------------------------------------------------------

move	Cut the selected file or folder in the file server.
------	-----------------------------------------------------

upload	Upload files to the current path or directory in the file system.
--------	-------------------------------------------------------------------

download	Downloads the file from the server and the multiple files can be downloaded as ZIP files.
----------	-------------------------------------------------------------------------------------------

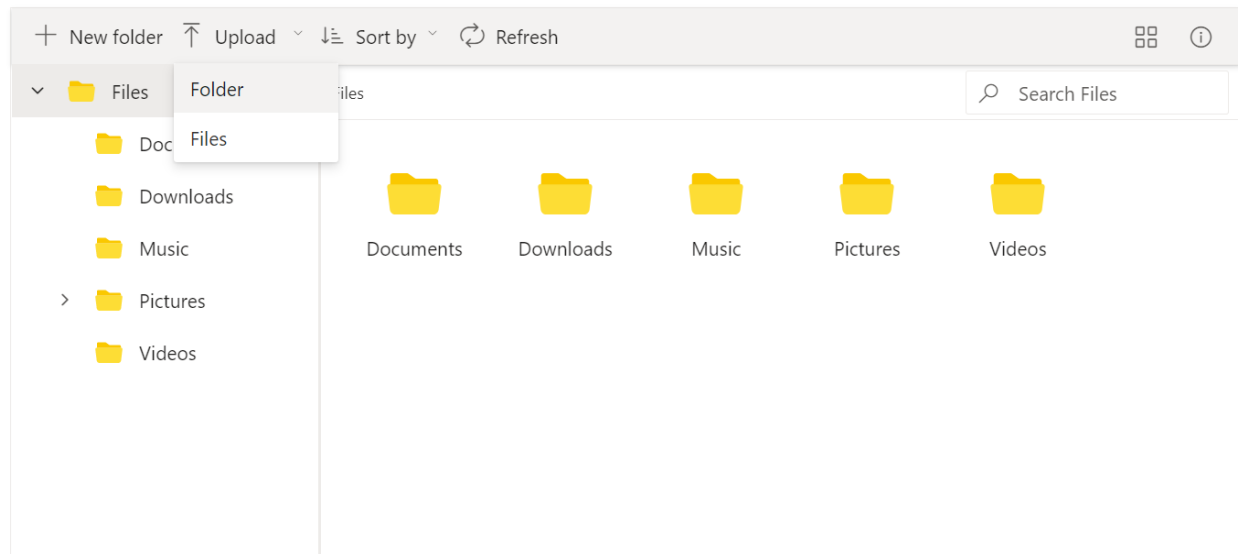
---

**Note:** The *CreateFolder*, *Remove*, and *Rename* actions will be reflected in the file manager only after the successful response from the server.

---

## Folder Upload support

To perform the directory(folder) upload in File Manager, set



`directoryUpload` property as true within the `uploadSettings` property. The directory upload feature is supported for the following file service providers:

- Physical file service provider.
- Azure file service provider.
- NodeJS file service provider.
- Amazon file service provider.

In the following example, directory upload is enabled/disabled on `DropDownButton` selection.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).AllowDragAndDrop(true).FileDragStart("onFileDragStart").FileDragStop("onF
 ileDragStop").FileDragging("onFileDragging").FileDropped("onFileDropped").Re
 nder()
 <!-- end of filemanager element -->
 </div>
</div>
<script>
```

```

function onFileDragStart() {
 console.log("Drag start");
}
function onFileDragStop() {
 console.log("Drag stop");
}
function onFileDragging() {
 console.log("File Dragging");
}
function onFileDropped() {
 console.log("File Dropped");
}
</script>

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 }
 }
 }
}

```

```

 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}

```

```

 }
 // Processing the Download operation
 public ActionResult Download(string downloadInput)
 {
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
 }
 public ActionResult Index()
 {
 return View();
 }
}
}

```

Output be like the below.

![[Directory upload](./images/directory-upload.png)]

#### Physical file service provider

To achieve the directory upload in the physical file service provider, use the below code snippet in `IActionResult Upload` method in the `Controllers/FileManagerController.cs` file.

```

`typescript
[Route("Upload")]
public IActionResult Upload(string path, IList<IFormFile> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 foreach (var file in uploadFiles)
 {
 var folders = (file.FileName).Split('/');
 // checking the folder upload
 if (folders.Length > 1)
 {
 for (var i = 0; i < folders.Length - 1; i++)
 {

```

```
string newDirectoryPath = Path.Combine(this.basePath + path, folders[i]);
if (!Directory.Exists(newDirectoryPath))
{
 this.operation.ToCamelCase(this.operation.Create(path, folders[i]));
}
path += folders[i] + "/";
}
}
}

uploadResponse = operation.Upload(path, uploadFiles, action, null);
if (uploadResponse.Error != null)
{
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = Convert.ToInt32(uploadResponse.Error.Code);
 Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
 uploadResponse.Error.Message;
}
return Content("");
}
`
```

Refer to the [GitHub](#) for more details

And also add the below code snippet in `FileManagerResponse Upload` method in `Models/PhysicalFileProvider.cs` file.

```
`typescript
string[] folders = name.Split('/');
string fileName = folders[folders.Length - 1];
var fullName = Path.Combine((this.contentRootPath + path), fileName);
`
```

Refer to the [GitHub](#) for more details.

#### *Azure file service provider*

For Azure file service provider, no customizations are needed for directory upload with server side and this will work with the below default upload method code.

Refer to the [GitHub](#) for more details.



*NodeJS file service provider*

To perform the directory upload in the NodeJS file service provider, use the below code snippet in `app.post` method in the `filesystem-server.js` file.

```
`typescript
var folders = (req.body.filename).split('/');
var filepath = req.body.path;
var uploadedFileName = folders[folders.length - 1];
// checking the folder upload
if (folders.length > 1)
{
for (var i = 0; i < folders.length - 1; i++)
{
var newDirectoryPath = path.join(contentRootPath + filepath, folders[i]);
if (!fs.existsSync(newDirectoryPath)) {
fs.mkdirSync(newDirectoryPath);
(async () => {
await FileManagerDirectoryContent(req, res, newDirectoryPath).then(data => {
response = { files: data };
response = JSON.stringify(response);
});
})();
}
filepath += folders[i] + "/";
}
fs.rename('./' + uploadedFileName, path.join(contentRootPath, filepath + uploadedFileName), function
(err) {
if (err) {
if (err.code !== 'EBUSY') {
errorValue.message = err.message;
errorValue.code = err.code;
}
}
});
}
```

`

Refer to the [GitHub](#) for more details.

#### *Amazon file service provider*

To perform the directory upload in the Amazon file service provider, use the below code snippet in `IActionResult AmazonS3Upload` method in the `Controllers/AmazonS3ProviderController.cs` file.

```
`typescript
foreach (var file in uploadFiles)
{
 var folders = (file.FileName).Split('/');
 // checking the folder upload
 if (folders.Length > 1)
 {
 for (var i = 0; i < folders.Length - 1; i++)
 {
 if (!this.operation.checkFileExist(path, folders[i]))
 {
 this.operation.ToCamelCase(this.operation.Create(path, folders[i], dataObject));
 }
 path += folders[i] + "/";
 }
 }
}
```

`

Refer to the [GitHub](#) for more details.

And also add the below code snippet in `AsyncUpload` method in `Models/AmazonS3FileProvider.cs` file.

```
`typescript
string[] folders = file.FileName.Split('/');
string name = folders[folders.Length - 1];
`
```

Refer to the [GitHub](#) for more details.

#### *File operation request and response Parameters*

The default parameters available in file operation request from the file manager and the corresponding response parameters required by the file manager are listed as follows.

### Read

The following table represents the request parameters of *read* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	read	Name of the file operation.
path	String	-	Relative path from which the data has to be read.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
data	FileManagerDirectoryContent	-	Details about the current path (directory).

*\*Refer [File request and response contents](#) for the contents of data\*.*

*Example:*

```
`typescript
{
 action: "read",
 path: "/",
 showHiddenItems: false,
 data: []
}
```

The following table represents the response parameters of *read* operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent	-	Details of files and folders present in given path or directory.
error	ErrorDetails	-	Error Details

*\*Refer [File request and response contents](#) for the contents of cwd, files, and error\*.*

*Example:*

```
`typescript
{
 cwd:
 {
 name:"Download",
 size:0,
 dateModified:"2019-02-28T03:48:19.8319708+00:00",
```

```
dateCreated:"2019-02-27T17:36:15.812193+00:00",
hasChild:false,
isFile:false,
type:"",
filterPath:"\\Download\\"
},
files:[
{
name:"Sample Work Sheet.xlsx",
size:6172,
dateModified:"2019-02-27T17:23:50.9651206+00:00",
dateCreated:"2019-02-27T17:36:15.8151955+00:00",
hasChild:false,
isFile:true,
type:".xlsx",
filterPath:"\\Download\\"
}
],
error:null,
details:null
}
`
```

### Create

The following table represents the request parameters of *create* operations.

Parameter	Type	Default	Explanation
action	String	create	Name of the file operation.
path	String	-	Relative path in which the folder has to be created.
name	String	-	Name of the folder to be created.
data	FileManagerDirectoryContent	-	Details about the current path (directory).

\*Refer [File request and response contents](#) for the contents of data\*

Example:

```
`typescript
```

```
{
 action: "create",
 data: [
 {
 dateCreated: "2019-02-27T17:36:15.6571949+00:00",
 dateModified: "2019-03-12T10:17:31.8505975+00:00",
 filterPath: "\",
 hasChild: true,
 isFile: false,
 name: files,
 nodeId: "fe_tree",
 size: 0,
 type: ""
 }
],
 name: "Hello",
 path: "/"
}
```

The following table represents the response parameters of *create* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the created folder
error	ErrorDetails	-	Error Details

*\*Refer [File request and response contents](#) for the contents of files and error\*.*

*Example:*

```
`typescript
{
 cwd: null,
 files: [
 {
 dateCreated: "2019-03-15T10:25:05.3596171+00:00",
 dateModified: "2019-03-15T10:25:05.3596171+00:00",
```

```
filterPath: null,
hasChild: false,
isFile: false,
name: "New",
size: 0,
type: ""
},
details: null,
error: null
},
`
```

### Rename

The following table represents the request parameters of *rename* operations.

Parameter	Type	Default	Explanation
action	String	rename	Name of the file operation.
path	String	-	Relative path in which the item is located.
name	String	-	Current name of the item to be renamed.
newname	String	-	New name for the item.
data	FileManagerDirectoryContent	-	Details of the item to be renamed.

*\*Refer [File request and response contents](#) for the contents of data\*.*

### Example:

```
`typescript
{
 action: "rename",
 data: [
 {
 dateCreated: "2019-03-20T05:22:34.621Z",
 dateModified: "2019-03-20T08:45:56.000Z",
 filterPath: "\\Pictures\\Nature\\",
 hasChild: false,
 iconClass: "e-fe-image",
```

```
isFile: true,
name: "seaviews.jpg",
size: 95866,
type: ".jpg"
},
newname: "seaview.jpg",
name: "seaviews.jpg",
path: "/Pictures/Nature/"
},
,
```

The following table represents the response parameters of *rename* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the renamed item.
error	ErrorDetails	-	Error Details

*\*Refer [File request and response contents](#) for the contents of files and error\*.*

*Example:*

```
`typescript
{
 cwd:null,
 files:[
 {
 name:"seaview.jpg",
 size:95866,
 dateModified:"2019-03-20T08:45:56+00:00",
 dateCreated:"2019-03-20T05:22:34.6214847+00:00",
 hasChild:false,
 isFile:true,
 type:".jpg",
 filterPath:"\\Pictures\\Nature\\seaview.jpg"
 }
],
}
```

```
error:null,
details:null
}
`
```

### Delete

The following table represents the request parameters of *delete* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	delete	Name of the file operation.
path	String	-	Relative path where the items to be deleted are located.
names	String[]	-	List of the items to be deleted.
data	FileManagerDirectoryContent	-	Details of the item to be deleted.

*\*Refer [File request and response contents](#) for the contents of data\*.*

*Example:*

```
`typescript
{
 action: "delete",
 path: "/Hello/",
 names: ["New"],
 data: []
}
`
```

The following table represents the response parameters of *delete* operations.

Parameter	Type	Default	Explanation
----	----	----	----
files	FileManagerDirectoryContent[]	-	Details about the deleted item(s).
error	ErrorDetails	-	Error Details

*\*Refer [File request and response contents](#) for the contents of files and error\*.*

*Example:*

```
`typescript
{
 cwd: null,
 details: null,
}
```



```
error: null,
files: [
{
 dateCreated: "2019-03-15T10:13:30.346309+00:00",
 dateModified: "2019-03-15T10:13:30.346309+00:00",
 filterPath: "\\Hello\\folder",
 hasChild: true,
 isFile: false,
 name: "folder",
 size: 0,
 type: ""
}
]
}
```

#### Details

The following table represents the request parameters of *details* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	details	Name of the file operation.
path	String	-	Relative path where the items are located.
names	String[]	-	List of the items to get details.
data	FileManagerDirectoryContent	-	Details of the selected item.

*\*Refer [File request and response contents](#) for the contents of data\*.*

*Example:*

```
`typescript
{
 action: "details",
 path: "/FileContents/",
 names: ["All Files"],
 data: []
}
```

The following table represents the response parameters of *details* operations.

Parameter	Type	Default	Explanation
----	----	----	----
details	FileManagerDirectoryContent	-	Details of the requested item(s).
error	ErrorDetails	-	Error Details

*\*Refer [File request and response contents](#) for the contents of details and error\*.*

*Example:*

```
`typescript
{
 cwd:null,
 files:null,
 error:null,
 details:
 {
 name:"All Files",
 location:"\\Files\\FileContents\\All Files",
 isFile:false,
 size:"679.8 KB",
 created:"3/8/2019 10:18:37 AM",
 modified:"3/8/2019 10:18:39 AM",
 multipleFiles:false
 }
}
```

### *Search*

The following table represents the request parameters of *search* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	search	Name of the file operation.
path	String	-	Relative path to the directory where the files should be searched.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
caseSensitive	Boolean	-	Defines search is case sensitive or not.
searchString	String	-	String to be searched in the directory.

|data|FileManagerDirectoryContent|-|Details of the searched item.|

*Example:*

```
`typescript
{
 action: "search",
 path: "/",
 searchString: "nature",
 showHiddenItems: false,
 caseSensitive: false,
 data: []
}
```

The following table represents the response parameters of *search* operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------

----	----	----	----
------	------	------	------

cwd FileManagerDirectoryContent - Path (Current Working Directory) details.
-----------------------------------------------------------------------------

files FileManagerDirectoryContent[] - Files and folders in the searched directory that matches the search input.
------------------------------------------------------------------------------------------------------------------

error ErrorDetails - Error Details
------------------------------------

*\*Refer [File request and response contents](#) for the contents of cwd, files and error.\**

*Example:*

```
`typescript
{
 cwd:
 {
 name:files,
 size:0,
 dateModified:"2019-03-15T10:07:00.8658158+00:00",
 dateCreated:"2019-02-27T17:36:15.6571949+00:00",
 hasChild:true,
 isFile:false,
 type:"",
 filterPath:"\\"
 }
}
```

```
},
files:[
{
name:"Nature",
size:0,
dateModified:"2019-03-08T10:18:42.9937708+00:00",
dateCreated:"2019-03-08T10:18:42.5907729+00:00",
hasChild:true,
isFile:false,
type:"",
filterPath:"\\FileContents\\Nature"
}
],
error:null,
details:null
}
`
```

### Copy

The following table represents the request parameters of *copy* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	copy	Name of the file operation.
path	String	-	Relative path to the directory where the files should be copied.
names	String[]	-	List of files to be copied.
targetPath	String	-	Relative path where the items to be pasted are located.
data	FileManagerDirectoryContent	-	Details of the copied item.
renameFiles	String[]	-	Details of the renamed item.

### Example:

```
`typescript
{
action: "copy",
path: "/",
names: ["6.png"],
```

```
renameFiles: ["6.png"],
targetPath: "/Videos/"
},
,
```

The following table represents the response parameters of *copy* operations.

Parameter	Type	Default	Explanation
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of copied files or folders
error	ErrorDetails	-	Error Details

*\*Refer [File request and response contents](#) for the contents of cwd, files and error.\**

*Example:*

```
`typescript
{
 cwd:null,
 files:[
 {
 path:null,
 action:null,
 newName:null,
 names:null,
 name:"justin.mp4",
 size:0,
 previousName:"album.mp4",
 dateModified:"2019-06-21T06:58:32+00:00",
 dateCreated:"2019-06-24T04:22:14.6245618+00:00",
 hasChild:false,
 isFile:true,
 type:".mp4",
 id:null,
 filterPath:"\\"
 }
],
```

```
error:null,
details:null
}
`
```

### Move

The following table represents the request parameters of *move* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	move	Name of the file operation.
path	String	-	Relative path to the directory where the files should be copied.
names	String[]	-	List of files to be moved.
targetPath	String	-	Relative path where the items to be pasted are located.
data	FileManagerDirectoryContent	-	Details of the moved item.
renameFiles	String[]	-	Details of the renamed item.

*Example:*

```
`typescript
{
 action: "move",
 path: "/",
 names: ["6.png"],
 renameFiles: ["6.png"],
 targetPath: "/Videos/"
}
`
```

The following table represents the response parameters of *copy* operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of cut files or folders
error	ErrorDetails	-	Error Details

\*Refer [File request and response contents](#) for the contents of *cwd*, *files* and *error*.\*

*Example:*

```
`typescript
```

```
{
 cwd:null,
 files:[
 {
 path:null,
 action:null,
 newName:null,
 names:null,
 name:"justin biber.mp4",
 size:0,
 previousName:"justin biber.mp4",
 dateModified:"2019-06-21T06:58:32+00:00",
 dateCreated:"2019-06-24T04:26:49.2690476+00:00",
 hasChild:false,
 isFile:true,
 type:".mp4",
 id:null,
 filterPath:"\\Videos\\"
 }
],
 error:null,
 details:null
}
```

### Upload

The following table represents the request parameters of *Upload* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	Save	Name of the file operation.
path	String	-	Relative path to the location where the file has to be uploaded.
uploadFiles	IList<IFormFile>	-	File that are uploaded.

*Example:*

```
`typescript
```

```
uploadFiles: (binary),
path: /,
action: Save,
data: {
 path:null,
 action:null,
 newName:null,
 names:null,
 name:"Downloads",
 size:0,
 previousName:null,
 dateModified:"2019-07-22T11:23:46.7153977 00:00",
 dateCreated:"2019-07-22T11:26:13.9047229 00:00",
 hasChild:false,
 isFile:false,
 type:"",
 id:null,
 filterPath:"\\",
 targetPath:null,
 renameFiles:null,
 uploadFiles:null,
 caseSensitive:false,
 searchString:null,
 showHiddenItems:false,
 fmiconClass:null,
 fmid:"fetree1",
 fmpId:null,
 fmselected:false,
 fmicon:null,
 data:null,
 targetData:null,
 permission:null
}
```



,

The upload response is an empty string.

Download

The following table represents the request parameters of *download* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	download	Name of the file operation
path	String	-	Relative path to location where the files to download are present.
names	String[]	-	Name list of the items to be downloaded.
data	FileManagerDirectoryContent	-	Details of the download item.

Example:

```
`typescript
{
 action:"download",
 path:"/",
 names:["1.png"],
 data:[
 {
 path:null,
 action:null,
 newName:null,
 names:null,
 name:"1.png",
 size:49792,
 previousName:null,
 dateModified:"2019-07-22T12:15:45.0972405+00:00",
 dateCreated:"2019-07-22T12:15:45.0816042+00:00",
 hasChild:false,
 isFile:true,
 type:".png",
 id:null,
 filterPath:"\\",
 targetPath:null,
```

```
renameFiles:null,
uploadFiles:null,
caseSensitive:false,
searchString:null,
showHiddenItems:false,
fmiconClass:"e-fe-image",
fmid:null,
fmpld:null,
fmselected:false,
fmicon:null,
data:null,
targetData:null,
permission:null,
fmcreated:"2019-07-22T12:15:45.081Z",
fmmodified:"2019-07-22T12:15:45.097Z",
fmimageUrl:"https://ej2-aspcore-service.azurewebsites.net/api/FileManager/GetImage?path=/1.png",
fmimageAttr:
{
 alt:"1.png"
},
fmhtmlAttr:
{
 class:"e-large-icon",
 title:"Employee.png"
}
]
}
```

Downloads the requested items from the file server in response.

#### [GetImage](#)

The following table represents the request parameters of *GetImage* operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------

```
|----|----|----|----|
```

```
|path|String|-|Relative path to the image file|
```

Return the image as a file stream in response.

The request from the file manager can be customized using the `beforeSend` event. Additional information can be passed to the file manager in file operation response and can be used in customization.

#### File request and response contents

The following table represents the contents of *data*, *cwd*, and *files* in the file manager request and response.

```
|Parameter|Type|Default|Explanation|
```

```
|----|----|----|----|
```

```
|name|String|-|File name|
```

```
|dateCreated|String|-|Date in which file was created (UTC Date string).|
```

```
|dateModified|String|-|Date in which file was last modified (UTC Date string).|
```

```
|filterPath|String|-|Relative path to the file or folder.|
```

```
|hasChild|Boolean|-|Defines this folder has any child folder or not.|
```

```
|isFile|Boolean|-|Say whether the item is file or folder.|
```

```
|size|Number|-|File size|
```

```
|type|String|-|File extension|
```

The following table represents the contents of *error* in the file manager request and response.

```
|Parameter|Type|Default|Explanation|
```

```
|----|----|----|----|
```

```
|code|String|-|Error code|
```

```
|message|String|-|Error message|
```

```
|fileExists|String[]|-|List of duplicate file names|
```

The following table represents the contents of *details* in the file manager request and response.

```
|Parameter|Type|Default|Explanation|
```

```
|----|----|----|----|
```

```
|name|String|-|File name|
```

```
|dateCreated|String|-|Date in which file was created (UTC Date string).|
```

```
|dateModified|String|-|Date in which file was last modified (UTC Date string).|
```

```
|filterPath|String|-|Relative path to the file or folder.|
```

```
|hasChild|Boolean|-|Defines this folder has any child folder or not.|
```

```
|isFile|Boolean|-|Say whether the item is file or folder.|
```

|size|Number|-|File size|

|type|String|-|File extension|

|multipleFiles|Boolean|-|Say whether the details are about single file or multiple files.|

### Action Buttons

The file manager has several menu buttons to access the file operations. The list of menu buttons available in the file manager is given in the following table.

|Menu Button|Behaviour|

|----|----|

|SortBy| Opens the sub menu to choose the sorting order and sorting parameter. |

|View| Opens the sub menu to choose the View. |

|Open| Navigates to the selected folder. Opens the preview for image files. |

|Refresh| Initiates the read operation for the current directory and displays the updated directory content. |

|NewFolder| Opens the new folder dialog box to receive the name for the new folder. |

|Rename| Opens the rename dialog box to receive the new name for the selected item. |

|Delete| Opens the delete dialog box to confirm the removal of the selected items from the file system. |

|Upload| Opens the upload box to select the items to upload to the file system. |

|Download| Downloads the selected item(s). |

|Details| Get details about the selected items and display them in details dialog box. |

|SelectAll| Selects all the files and folders displayed in the view section. |

The action menu buttons are present in the toolbar and context menu. The toolbar contains the buttons based on the selected items count, while the context menu will appear with a list based on the target.

### Toolbar

The toolbar can be divided into two sections as right and left. Whenever the toolbar buttons exceed the size, the buttons present in the left section of the toolbar will be moved to the toolbar popup.

The following table provides the toolbar buttons that appear based on the selection.

<!-- markdownlint-disable MD033 -->

Selected Items Count	Left section	Right section
0 (none of the item )	<i>SortBy</i> Refresh <i>NewFolder</i> Upload	<i>View</i> Details
1 (single item selected)	<i>Delete</i> Download* Rename	<i>Selected items count</i> View* Details
>1 (multiple selection)	<i>Delete</i> Download	<i>Selected items count</i> View* Details

### Context menu

The following table provides the default context menu item and the corresponding target areas.

<!-- markdownlint-disable MD033 -->

Menu Name	Menu Items	Target
Layout	<i>SortBy View Refresh NewFolder Upload</i> Details* Select all	<i>Empty space in the view section (details view and large icon view area). Empty folder content.</i>
Folders	<i>Open Delete Rename</i> Downloads* Details	* Folders in treeview, details view, and large icon view.
Files	<i>Open Delete Rename</i> Downloads* Details	* Files in details view and large icon view.

## Views

View is the section where the files and folders are displayed for the user to browse. The [view](#) API can also be used to change the initial view of the file manager.

The file manager has two types of [views](#) to display the files and folders.

- [Largelcons View](#)
- [Details View](#)

## Largelcons View

By Default, File Manager is rendered with largeicons view. The following example demonstrate this.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
```

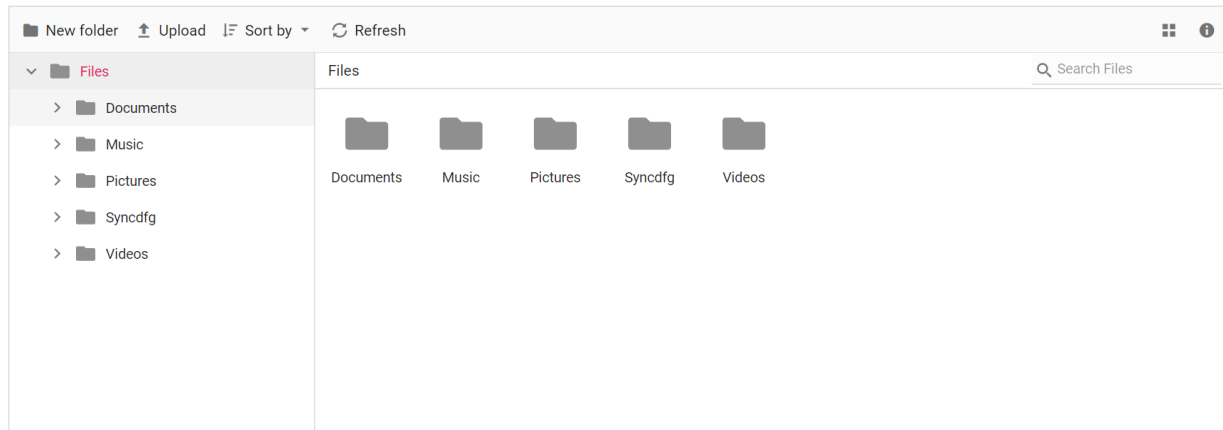
```
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 }
 }
 }
}
```

```

 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



### Details View

The default appearance of the file manager can be changed from largeicons to details view by using [view](#) property. In the Details view, the files are displayed in a sorted list order. This file list comprises of several columns of information about the files such as **Name**, **Date Modified**, **Type**, and **Size**. The following example demonstrate the file manager with details view.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).View(Syncfusion.EJ2.FileManager.ViewType.Details).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
```



```

{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder

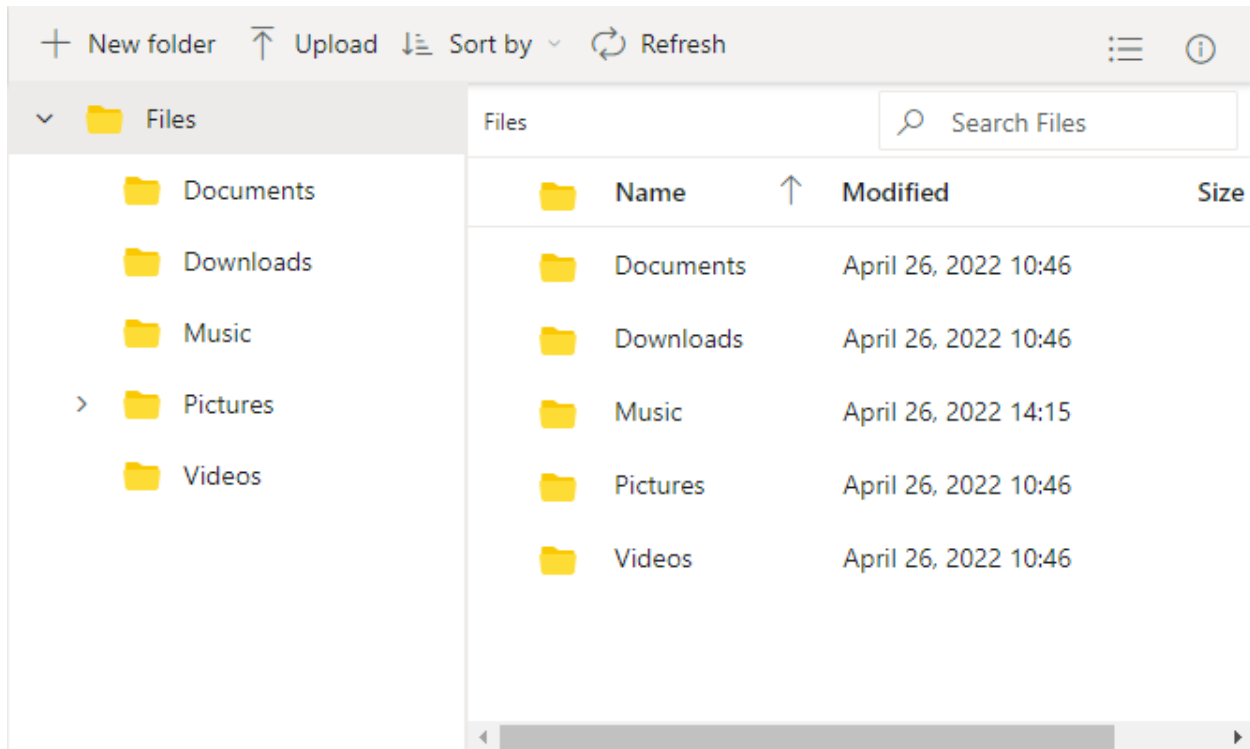
```

```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



### Customizing File Manager functionalities

The file manager component allows customizing its functionalities like, context menu, searching, uploading, toolbar using APIs. Given below are some of the functionalities that can be customized in the File Manager,

- [Context menu customization](#)
- [Details view customization](#)
- [Navigation pane customization](#)
- [Show/Hide file extension](#)
- [Show/Hide hidden items](#)
- [Show/Hide thumbnail images in large icons view](#)
- [Toolbar customization](#)
- [Upload customization](#)
- [Tooltip customization](#)

#### Context menu customization

The context menu settings like, items to be displayed on files, folders and layout click and visibility can be customized using [contextMenuSettings](#) property.

#### CSHTML

```
@{
 string[] files = new string[] { "Open", "|", "Details" };
 string[] folder = new string[] { "Open", "|", "Details" };
 string[] layout = new string[] { "SortBy", "View", "Refresh", "|",
 "Details" };
}
<div class="control-section">
```

```

<div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).ContextMenuSettings(new
Syncfusion.EJ2.FileManager.FileManagerContextMenuSettings()
 {
 File = files,
 Folder = folder,
 Layout = layout
 }).Render()
 <!-- end of filemanager element -->
</div>
</div>

```

### HOMECONTROLLER\_MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items

```

```

 return
 }
 JsonResult(operation.ToCamelCase(operation.GetFiles(args.Path,
args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 }
 JsonResult(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 }
 JsonResult(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 }
 JsonResult(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 JsonResult(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)

```

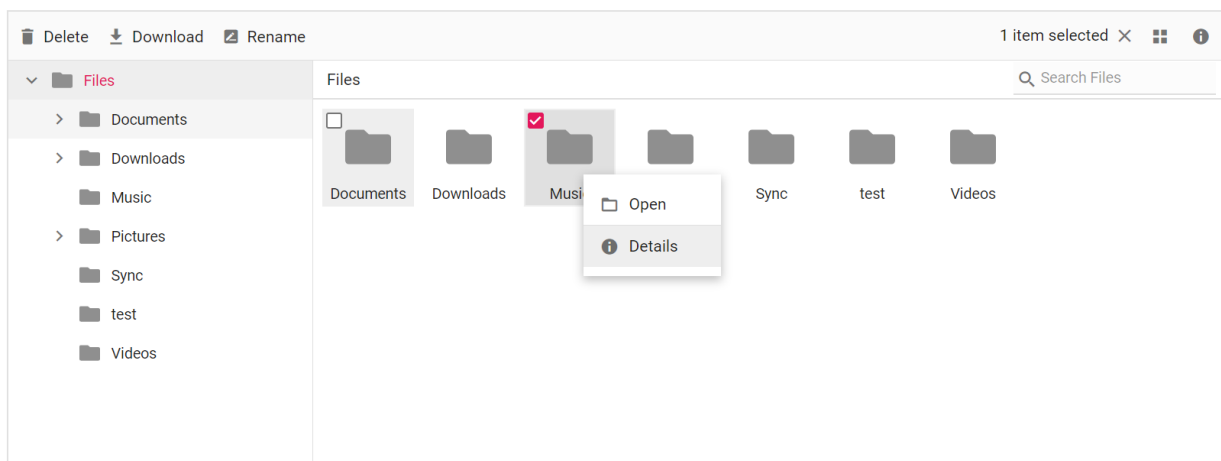
```

 {
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);

 return Content("");
 }
 // Processing the Download operation
 public ActionResult Download(string downloadInput)
 {
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
 }
 public ActionResult Index()
 {
 return View();
 }
}
}

```

Output be like the below.



### Details view customization

The details view settings like, column width, header text, template for each field can be customized using [detailsViewSettings](#) property.

**CSHTML**

```

<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->

@Html.EJS().FileManager("filemanager").View(Syncfusion.EJ2.FileManager.ViewType.Details).AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).DetailsViewSettings(new
Syncfusion.EJ2.FileManager.FileManagerDetailsViewSettings()
 {
 Columns = new List<Syncfusion.EJ2.FileManager.FileManagerColumn>
 {
 new Syncfusion.EJ2.FileManager.FileManagerColumn
 {
 Field = "name",
 HeaderText = "File Name",
 Width = "auto",
 Template = "${name}",
 CustomAttributes = new Dictionary<string, string>
 {
 { "class", "e-fe-grid-name" }
 }
 },
 new Syncfusion.EJ2.FileManager.FileManagerColumn
 {
 Field = "size",
 HeaderText = "File Size",
 Template = "${size}"
 },
 new Syncfusion.EJ2.FileManager.FileManagerColumn
 {
 Field = "_fm_modified",
 HeaderText = "Date Modified",
 }
 }
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>

```

**HOMECONTROLLER MVC.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package

```

```

using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders

```



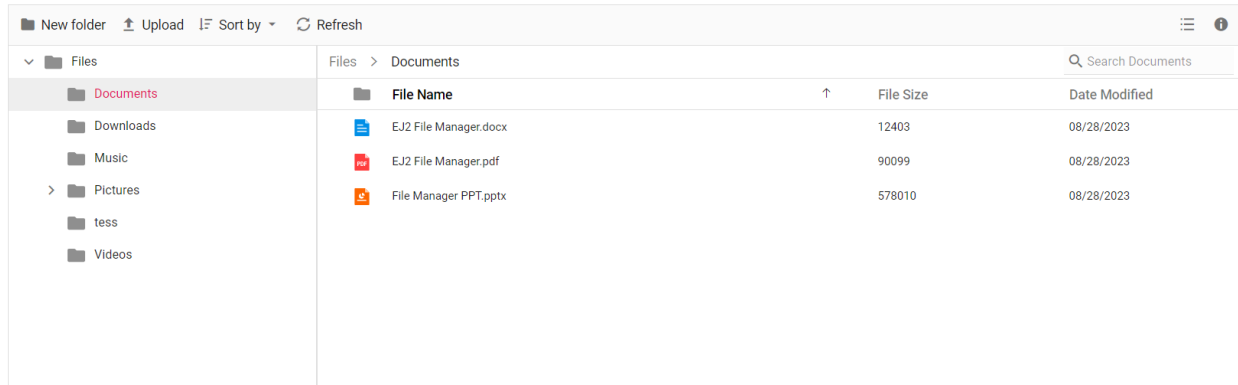
```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 // SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 // which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 // file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 // uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 // Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}

```

```
}
}
```

Output be like the below.



### Navigation pane customization

The navigation pane settings like, minimum and maximum width and visibility can be customized using [navigationPaneSettings](#) property.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).NavigationPaneSettings(new
 Syncfusion.EJ2.FileManager.FileManagerNavigationPaneSettings()
 {
 MaxWidth = "850px",
 MinWidth = "140px"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
```

```

// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 }
 }
 }
}

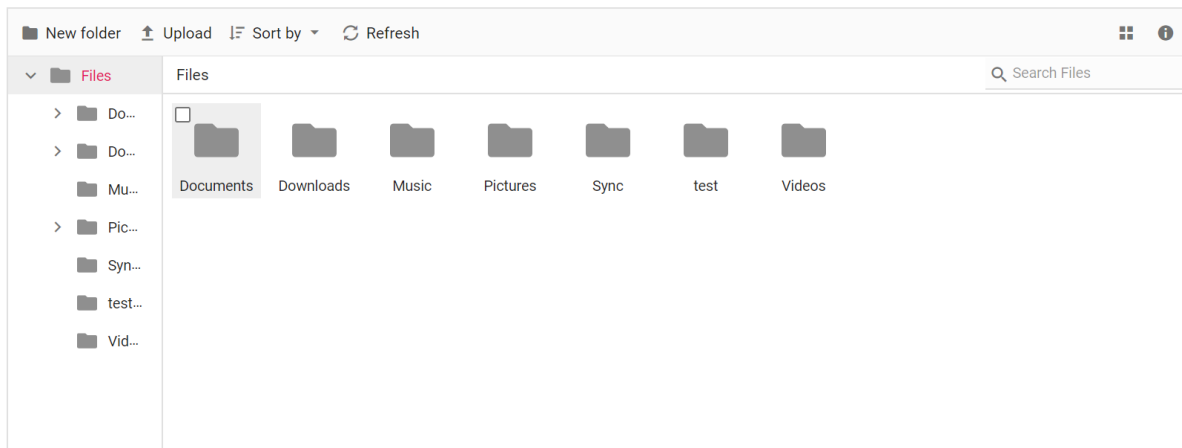
```

```

 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



### Show/Hide file extension

The file extensions are displayed in the File Manager by default. This can be hidden by disabling the [showFileExtension](#) property.

In File Manager [fileLoad](#) and [fileOpen](#) events are triggered before the file/folder is rendered and before the file/folder is opened respectively. These events can be utilized to perform operations before a file/folder is rendered or opened.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).ShowFileExtension(false).FileLoad("onBeforeFileLoad").FileOpen("onBeforeFileOpen").Render()
 <!-- end of filemanager element -->
 </div>
</div>
<script>
 // File Manager's file beforeFileLoad function
 function onBeforeFileLoad(args) {
 console.log(args.fileDetails.name + " is loading");
 }
 // File Manager's file beforeFileOpen function
 function onBeforeFileOpen(args) {
 console.log(args.fileDetails.name + " is opened");
 }
</script>
```

**HOMECONTROLLER MVC.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file

```

```

 return
 }
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}

// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}

// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}

// Processing the GetImage operation

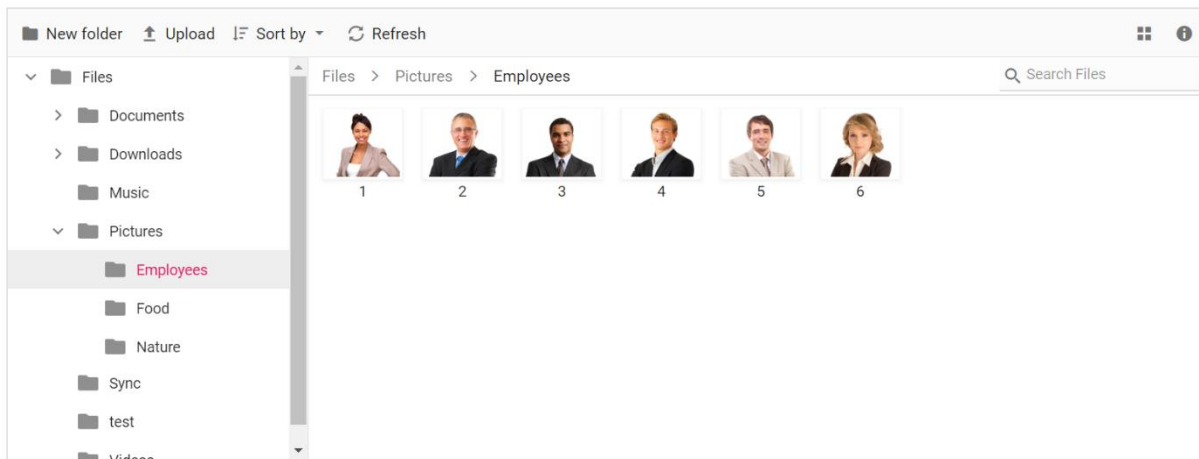
```

```

public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



### Show/Hide hidden items

The File Manager provides support to show/hide the hidden items by enabling/disabling the [showHiddenItems](#) property.

### CSHTML

```

<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).ShowHiddenItems(true).Render()
 <!-- end of filemanager element -->
 </div>
</div>

```

### HOMECONTROLLER MVC.CS



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":

```

```

 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;

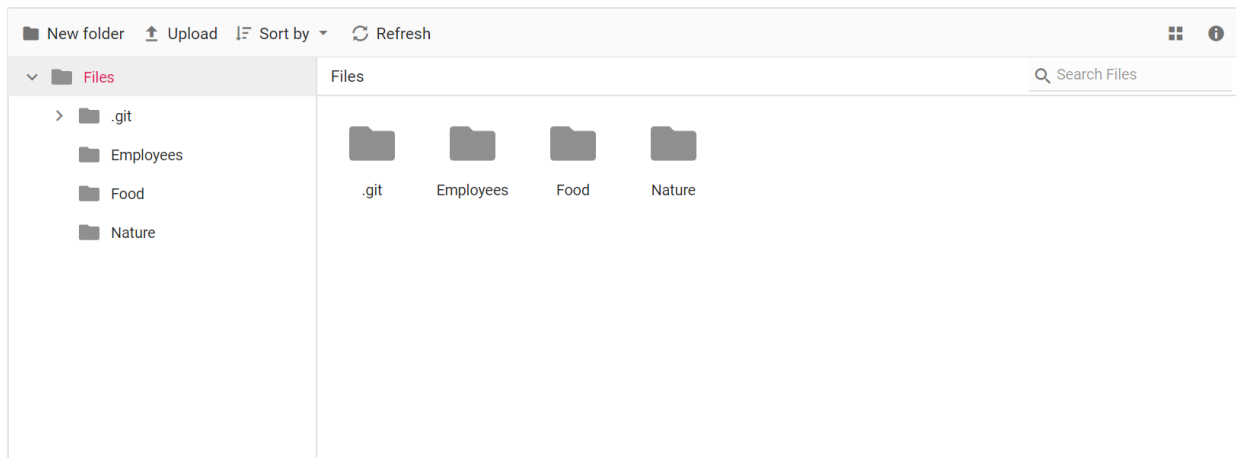
```

```

 return operation.GetImage(args.Path, args.Id, false, null,
null);
 }
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.



### Show/Hide thumbnail images in large icons view

The thumbnail images are displayed in the File Manager's large icons view by default. This can be hidden by disabling the [showThumbnail](#) property.

### CSHTML

```

<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).ShowThumbnail(false).Render()
 <!-- end of filemanager element -->
 </div>
</div>

```

### HOMECONTROLLER\_MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

```

```

//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 }
 }
 }
}

```

```

 // Path - Current path where details of file/folder is requested; Name - Names of the requested folders
 return
 }
 JsonResult(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed; SearchString - String typed in the searchbox; CaseSensitive - Boolean value which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString, args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name, args.NewName)));
}
return null;
}

// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}

// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names - Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}

// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()

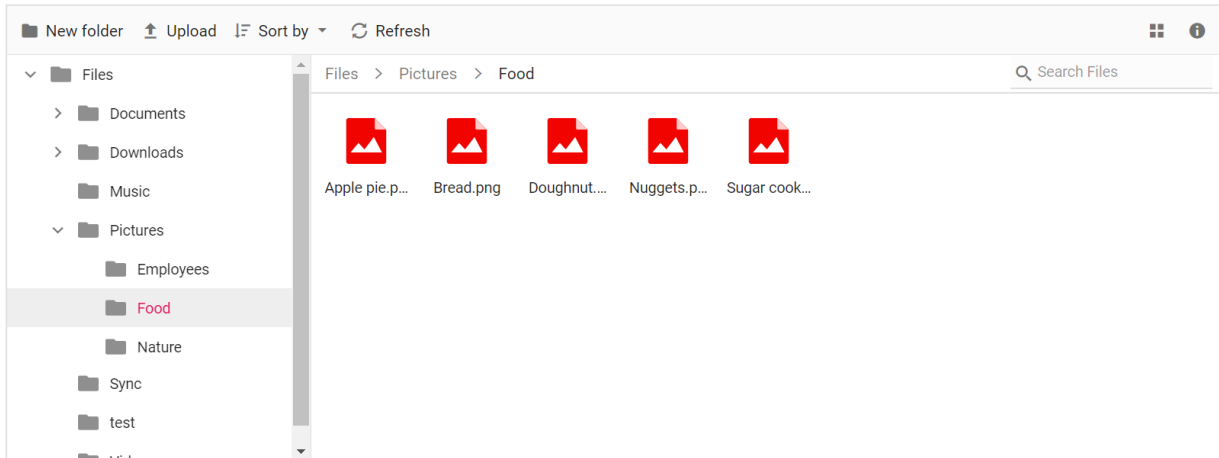
```

```

 {
 return View();
 }
}
}

```

Output be like the below.



### Toolbar customization

The toolbar settings like, items to be displayed in toolbar and visibility can be customized using [toolbarSettings](#) property.

### CSHTML

```

<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings()
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).ToolBarItems(new List<FileManagerToolBarItem> {
 new FileManagerToolBarItem { Name= "NewFolder" },
 new FileManagerToolBarItem { Name= "Upload" },
 new FileManagerToolBarItem { Name= "SortBy" },
 new FileManagerToolBarItem { Name= "Refresh" },
 new FileManagerToolBarItem { Name= "Cut" },
 new FileManagerToolBarItem { Name= "Copy" },
 new FileManagerToolBarItem { Name= "Paste" },
 new FileManagerToolBarItem { Name= "Delete" },
 new FileManagerToolBarItem { Name= "Download" },
 new FileManagerToolBarItem { Name= "Rename" },
 new FileManagerToolBarItem { Template= "#checkboxTemplate",
Name= "Select" },
 new FileManagerToolBarItem { Name= "Selection" },
 new FileManagerToolBarItem { Name= "View" },
 new FileManagerToolBarItem { Name= "Details" }
 })

```

```

 }).Render()
 <!-- end of filemanager element -->

@Html.EJS().CheckBox("checkboxTemplate").Change("onChange").Checked(false).Label("Select All").Render()
 </div>
</div>
<script>
 function onChange(args) {
 var filemanagerInstance =
document.getElementById("filemanager").ej2_instances[0];
 var checkboxInstance =
document.getElementById("checkboxTemplate").ej2_instances[0];
 if (args.checked) {
 filemanagerInstance.selectAll();
 checkboxInstance.label = 'Unselect All';
 }
 else {
 filemanagerInstance.clearSelection();
 checkboxInstance.label = 'Select All';
 }
 }
</script>

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":

```

```

 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 }
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 }
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 }
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 }
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation

```



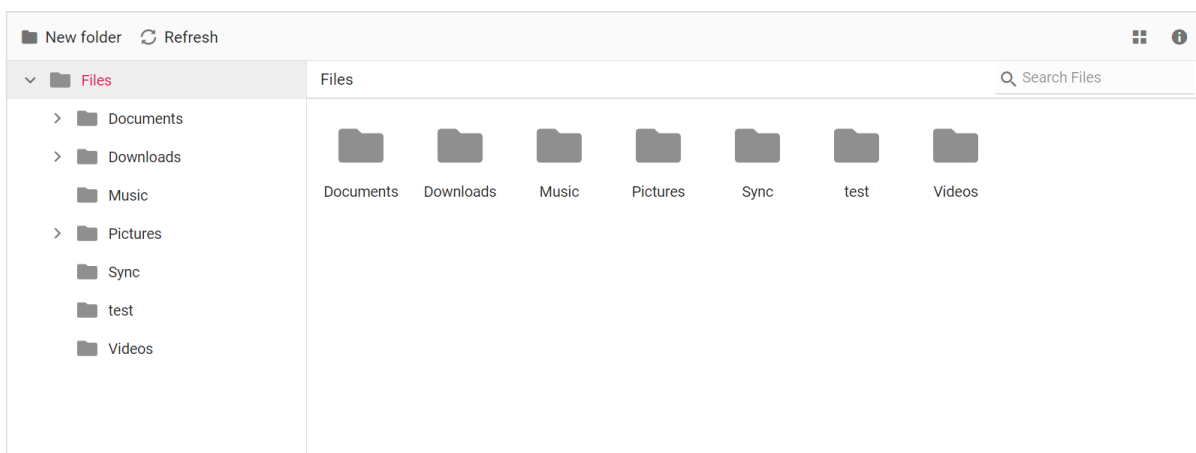
```

 public ActionResult Upload(string path,
 IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
 {
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);

 return Content("");
 }
 // Processing the Download operation
 public ActionResult Download(string downloadInput)
 {
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
 }
 public ActionResult Index()
 {
 return View();
 }
 }
}

```

Output be like the below.



See Also

- [How to add new items or customize default items](#)

### Upload customization

The upload settings like, minimum and maximum file size and enabling auto upload can be customized using [uploadSettings](#) property.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).UploadSettings(new
Syncfusion.EJ2.FileManager.FileManagerUploadSettings()
 {
 MaxFileSize = 233332,
 MinFileSize = 250,
 AutoUpload = true
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### HOMECONTROLLER\_MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 }
 }
}
```

```

 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":

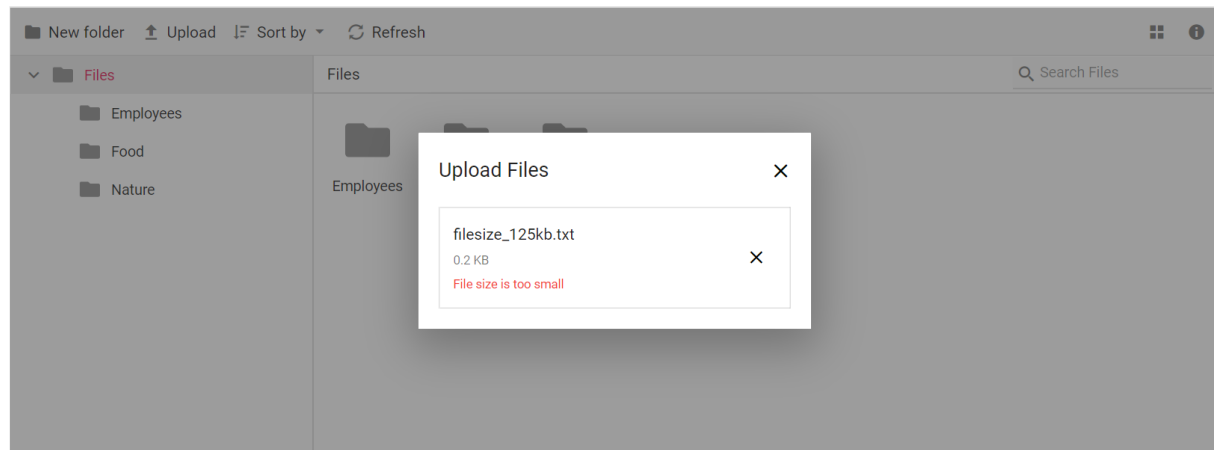
```

```

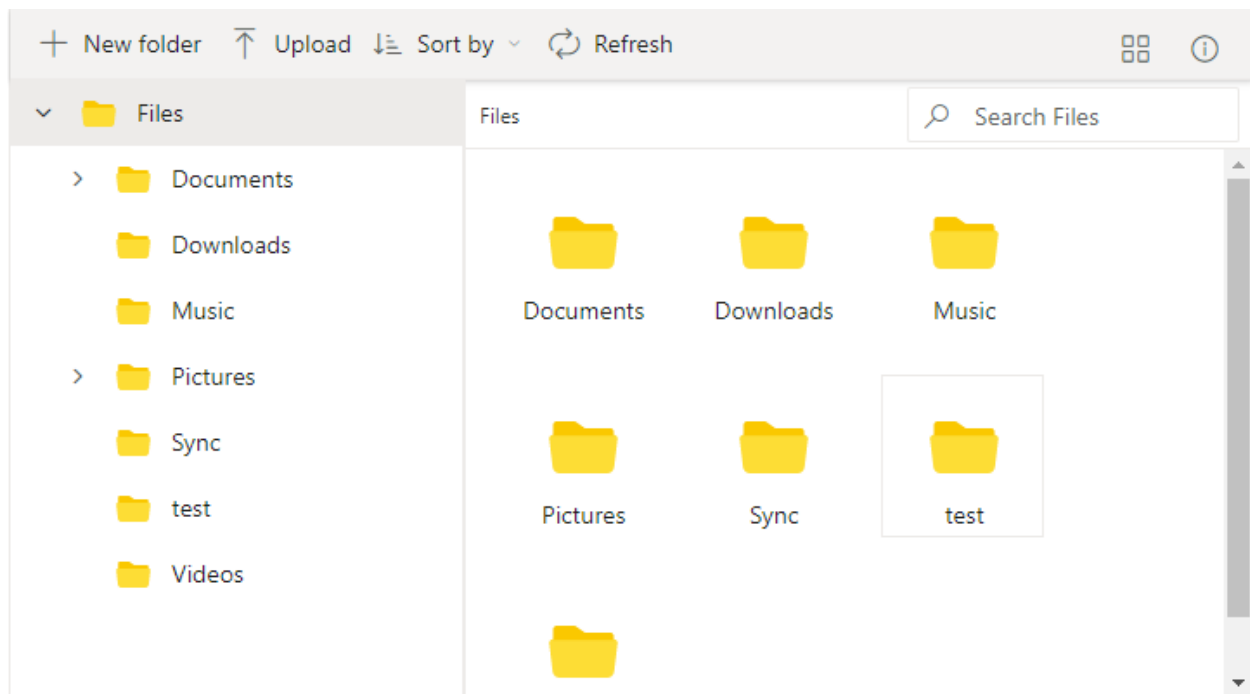
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below, when file size is below the `minFileSize`.



Output be like the below, when file size is above the `minFileSize`.



### Tooltip customization

The tooltip value can be customized by adding extra content to the title of the toolbar, navigation pane, details view and large icons of the file manager element.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("file").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
```

```

 DownloadUrl = "/Home/Download"
 }).Created("addTooltip").FileLoad("fileLoad").Render()
 <!-- end of filemanager element -->
</div>
</div>
<script>
 function addTooltip() {
 var fileObj = document.getElementById("file").ej2_instances[0];
 var tooltip = new ej.popups.Tooltip({
 target: '#' + fileObj.element.id + '_toolbar [title]',
 beforeRender: onTooltipBeforeRender
 });
 tooltip.appendTo('#' + fileObj.element.id + '_toolbar');
 }
 function onTooltipBeforeRender(args) {
 var fileObj =
document.getElementById("file").ej2_instances[0].element.id;
 var buttonId = args.target.childNodes[0].id;
 var description = '';
 switch (buttonId) {
 case fileObj + '_tb_newfolder':
 description = 'Create a new folder';
 break;
 case fileObj + '_tb_upload':
 description = 'Upload new files';
 break;
 case fileObj + '_tb_cut':
 description = 'Cut files and folders from the current
location';
 break;
 case fileObj + '_tb_copy':
 description = 'Copy files and folders from the current
location';
 break;
 case fileObj + '_tb_paste':
 description = 'Paste files and folders in the current
location';
 break;
 case fileObj + '_tb_delete':
 description = 'Delete selected files and folders';
 break;
 case fileObj + '_tb_download':
 description = 'Download selected files and folders';
 break;
 case fileObj + '_tb_rename':
 description = 'Rename selected file or folder';
 break;
 case fileObj + '_tb_sortby':
 description = 'Change the file sorting order';
 break;
 case fileObj + '_tb_refresh':
 description = 'Refresh the current location';
 break;
 case fileObj + '_tb_selection':
 description = 'Following items are currently selected:';
 for (var i = 0; i < fileObj.selectedItems.length; i++) {

```

```

 description = description + '</br>' +
fileObj.selectedItems[i];
 }
 break;
 case fileObj + '_tb_view':
 description = 'Switch the layout view';
 break;
 case fileObj + '_tb_details':
 description = 'Get the details of the seletced items';
 break;
 default:
 description = '';
 break;
 }
 this.content = args.target.getAttribute('title') + '</br>' +
description;
}
function fileLoad(args) {
 //Native tooltip customization to display additonal information in
new line
 var target = args.element;
 if (args.module === 'DetailsView') {
 var ele = select('[title]', args.element);
 var title = args.fileDetails.name +
 '\n' + args.fileDetails.dateModified;
 ele.setAttribute('title', title);
 }
 else if (args.module === 'LargeIconsView') {
 var title = args.fileDetails.name +
 '\n' + args.fileDetails.dateModified;
 target.setAttribute('title', title);
 }
}
</script>

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {

```

```

 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 // SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 // which specifies whether the search must be casesensitive

```

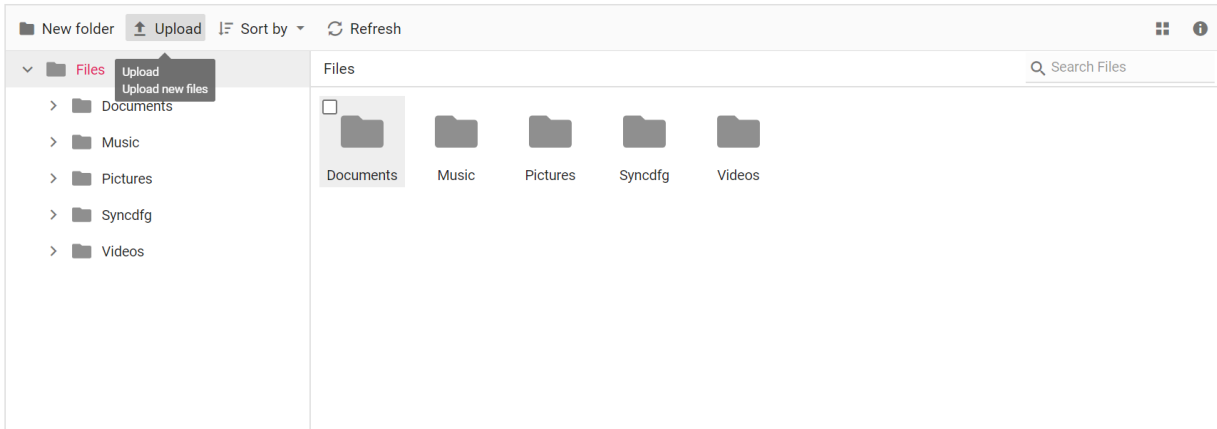


```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 // file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 // uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 // Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



## Multiple Selection

The file manager allows you to select multiple files by enabling the [allowMultiSelection](#) property (enabled by default). The multiple selection can be done by pressing the **Ctrl** key or **Shift** key and selecting the files. The check box can also be used to do multiple selection. **Ctrl + A** can be used to select all files in the current directory. The [fileSelect](#) event will be triggered when the items of file manager control is selected or unselected.

The below example shows File manager allows you to select multiple files using [allowMultiSelection](#) property.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).AllowMultiSelection(true).View(Syncfusion.EJ2.FileManager.ViewType.Detail
s).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
```

```

// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":

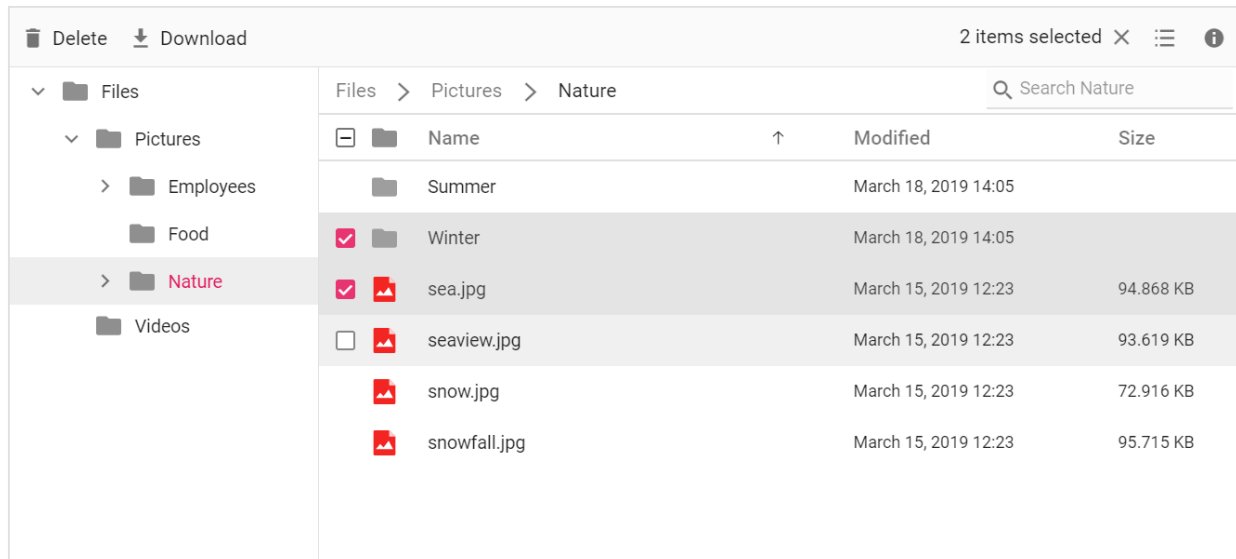
```

```

 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



## Drag And Drop

The file manager allows files or folders to be moved from one folder to another by using the [allowDragAndDrop](#) property. It also supports uploading a file by dragging it from Windows Explorer to File Manager control. You can enable or disable this support by using the [allowDragAndDrop](#) property of file manager.

The event triggered in drag and drop support are,

- [fileDragStart](#) - Triggers when the file/folder dragging is started.
- [fileDragging](#) - Triggers while dragging the file/folder.
- [fileDragStop](#) - Triggers when the file/folder is about to be dropped at the target.
- [fileDropped](#) - Triggers when the file/folder is dropped.

## CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).AllowDragAndDrop(true).FileDragStart("onFileDragStart").FileDragStop("onF
 ileDragStop").FileDragging("onFileDragging").FileDropped("onFileDropped").Re
 nder()
 <!-- end of filemanager element -->
 </div>
</div>
```

```
<script>
 function onFileDragStart() {
 console.log("Drag start");
 }
 function onFileDragStop() {
 console.log("Drag stop");
 }
 function onFileDragging() {
 console.log("File Dragging");
 }
 function onFileDropped() {
 console.log("File Dropped");
 }
</script>
```

### HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 JsonConvert.SerializeObject(operation.GetFiles(args.Path,
 args.ShowHiddenItems));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 }
 }
 }
}
```

```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 }
 JsonResult(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 }
 JsonResult(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 JsonResult(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)

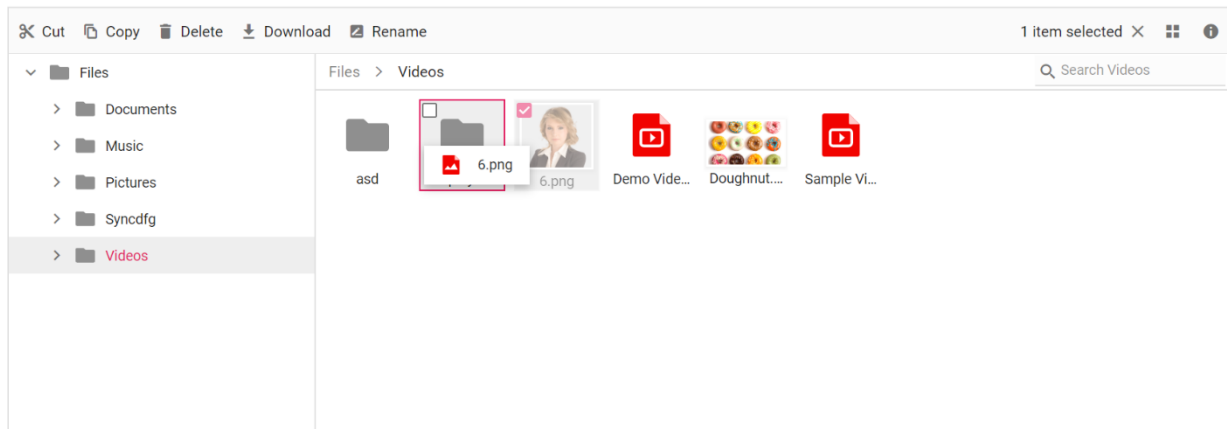
```

```

 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
 }
 // Processing the Download operation
 public ActionResult Download(string downloadInput)
 {
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
 }
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.



## File system provider

The file system provider allows the File Manager component to manage the files and folders in a physical or cloud-based file system. It provides the methods for performing various file actions like creating a new folder, copying and moving of files or folders, deleting, uploading, and downloading the files or folders in the file system.

The following file providers are added in Syncfusion EJ2 File Manager component.



- [ASP.NET Core file system provider](#)
- [ASP.NET MVC 5 file system provider](#)
- [ASP.NET Core Azure cloud file system Provider](#)
- [ASP.NET MVC 5 Azure cloud file system Provider](#)
- [Amazon S3 cloud file system provider](#)
- [File Transfer Protocol file system provider](#)
- [SQL database file system provider](#)
- [NodeJS file system provider](#)
- [Google Drive file system provider](#)
- [Firebase Realtime Database file system provider](#)
- [IBM Cloud Object Storage provider](#)

### ASP.NET Core file system provider

The ASP.NET Core file system provider allows the users to access and manage the physical file system. To get started, clone the [ej2-aspcore-file-provider](#) using the following command.

```
`typescript
```

```
git clone https://github.com/SyncfusionExamples/ej2-aspcore-file-provider ej2-aspcore-file-provider
cd ej2-aspcore-file-provider
`
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, set the root directory of the physical file system in the FileManager controller.

After setting the root directory of the file system, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the files in the physical file system.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with ASP.NET Core service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "http://localhost:{port}/api/FileManager/FileOperations",
 GetImageUrl =
"http://localhost:{port}/api/FileManager/GetImage",
 UploadUrl = "http://localhost:{port}/api/FileManager/Upload",
 DownloadUrl = "http://localhost:{port}/api/FileManager/Download"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### FILE-SYSTEM-PROVIDER.CS

---

**Note:** To learn more about file actions that can be performed with ASP.NET Core file system provider, refer to this [link](#)

---

### ASP.NET MVC 5 file system provider

The ASP.NET MVC5 file system provider allows the users to access and manage the physical file system. To get started, clone the [ej2-aspmvc-file-provider](#) using the following command.

```
`typescript
git clone https://github.com/SyncfusionExamples/ej2-aspmvc-file-provider ej2-aspmvc-file-provider
cd ej2-aspmvc-file-provider
`
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, set the root directory of the physical file system in the FileManager controller using the Root Folder method.

After setting the root directory of the file system, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the files in the physical file system.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with ASP.NET MVC service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "http://localhost:{port}/FileManager/FileOperations",
 GetImageUrl = "http://localhost:{port}/FileManager/GetImage",
 UploadUrl = "http://localhost:{port}/FileManager/Upload",
 DownloadUrl = "http://localhost:{port}/FileManager/Download"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### FILE-SYSTEM-MVC.CS

---

**Note:** To learn more about file actions that can be performed with ASP.NET MVC 5 file system provider, refer to this [link](#).

---

### ASP.NET Core Azure cloud file system Provider

In ASP.NET Core, Azure file system provider allows the users to access and manage the blobs in the Azure blob storage. To get started, clone the [azure-aspcore-file-provider](#) using the following command.

```
`typescript
git clone https://github.com/SyncfusionExamples/azure-aspcore-file-provider azure-aspcore-file-provider
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, register the Azure storage by passing details like name, password, and blob name to the Register Azure method in the FileManager controller.

```
`typescript
void RegisterAzure(string accountName, string accountKey, string blobName)
```

Then, set the blob container and the root blob directory by passing the corresponding URLs as parameters in the setBlobContainer method as follows.

```
`typescript
void setBlobContainer(string blobPath, string filePath)
```

---

**Note:** Also, assign the same *blobPath* URL and *filePath* URL in [AzureFileOperations](#) and [AzureUpload](#) methods in the FileManager controller to determine the original path of the Azure blob.

---

After setting the blob container references, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Azure blob storage.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with Azure service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url =
 "http://localhost:{port}/api/AzureProvider/AzureFileOperations",
 GetImageUrl =
 "http://localhost:{port}/api/AzureProvider/AzureGetImage",
 UploadUrl =
 "http://localhost:{port}/api/AzureProvider/AzureUpload",
 DownloadUrl =
 "http://localhost:{port}/api/AzureProvider/AzureDownload"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### AZURE-FILE-SYSTEM.CS

---

**Note:** To learn more about file actions that can be performed with ASP.NET Core Azure cloud file system provider, refer to this [link](#)

---

### ASP.NET MVC 5 Azure cloud file system Provider

In ASP.NET MVC, Azure file system provider allows the users to access and manage the blobs in the Azure blob storage. To get started, clone the [ej2-azure-aspmvc-file-provider](https://github.com/SyncfusionExamples/ej2-azure-aspmvc-file-provider) using the following command.

```
`typescript
git clone https://github.com/SyncfusionExamples/ej2-azure-aspmvc-file-provider ej2-azure-aspmvc-file-provider
`
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, register the Azure storage by passing details like name, password, and blob name to the Register Azure method in the FileManager controller.

```
`typescript
void RegisterAzure(string accountName, string accountKey, string blobName)
`
```

Then, set the blob container and the root blob directory by passing the corresponding URLs as parameters in the **setBlobContainer** method as follows.

```
`typescript
void setBlobContainer(string blobPath, string filePath)
`
```

---

**Note:** Also, assign the same *blobPath* URL and *filePath* URL in [AzureFileOperations](#) and [AzureUpload](#) methods in the FileManager controller to determine the original path of the Azure blob.

---

After setting the blob container references, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Azure blob storage.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with Azure service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url =
 "http://localhost:{port}/AzureProvider/AzureFileOperations",
 GetImageUrl =
 "http://localhost:{port}/AzureProvider/AzureGetImage",
 UploadUrl = "http://localhost:{port}/AzureProvider/AzureUpload",
 DownloadUrl =
 "http://localhost:{port}/AzureProvider/AzureDownload"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

## MVC-AZURE-FILE-SYSTEM.CS

**Note:** To learn more about file actions that can be performed with ASP.NET MVC 5 Azure cloud file system provider, refer to this [link](#).

### Amazon S3 cloud file system provider

In ASP.NET Core, Amazon **S3** (*Simple Storage Service*) cloud file system provider allows the users to access and manage a server hosted file system as collection of objects stored in the Amazon S3 Bucket. To get started, clone the [amazon-s3-aspcore-file-provider](#) using the following command.

```
`typescript
git clone https://github.com/SyncfusionExamples/amazon-s3-aspcore-file-provider.git amazon-s3-
aspcore-file-provider.git
`
```

**Note:** To learn more about creating and configuring an Amazon S3 bucket, refer to this [link](#).

After cloning, open the project in Visual Studio and restore the NuGet packages. Now, register Amazon S3 client account details like *awsAccessKeyId*, *awsSecretKeyId* and *awsRegion* details in **RegisterAmazonS3** method in the FileManager controller to perform the file operations.

```
`typescript
void RegisterAmazonS3(string bucketName, string awsAccessKeyId, string awsSecretAccessKey, string
bucketRegion)
`
```

After registering the Amazon client account details, just build and run the project. Now, the project will be hosted in <http://localhost:{port}> and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Amazon **S3** (*Simple Storage Service*) bucket's objects storage.

## CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with Amazon service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url =
"http://localhost:{port}/api/AmazonS3Provider/AmazonS3FileOperations",
 DownloadUrl =
"http://localhost:{port}/api/AmazonS3Provider/AmazonS3Download",
 UploadUrl =
"http://localhost:{port}/api/AmazonS3Provider/AmazonS3Upload",
 GetImageUrl =
"http://localhost:{port}/api/AmazonS3Provider/AmazonS3GetImage"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

```
</div>
</div>
```

## AMAZON-FILE-SYSTEM.CS

**Note:** To learn more about the file actions that can be performed with Amazon S3 Cloud File System provider, refer to this [link](#).

### File Transfer Protocol file system provider

In ASP.NET Core, File Transfer Protocol file system provider allows the users to access to the hosted file system as collection of objects stored in the file storage using File Transfer Protocol. To get started, clone the [ftp-aspcore-file-provider](#) using the following command.

```
`typescript
git clone https://github.com/SyncfusionExamples/ftp-aspcore-file-provider.git ftp-aspcore-file-
provider.git
`
```

After cloning, open the project in Visual Studio and restore the NuGet packages. Now, register File Transfer Protocol details like *hostName*, *userName* and *password* in **SetFTPConnection** method in the FileManager controller to perform the file operations.

```
`typescript
void SetFTPConnection(string hostName, string userName, string password)
`
```

After registering the File Transfer Protocol details, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows you to manage the FTP's objects storage.

## CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with FTP service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url =
"http://localhost:{port}/api/FTPProvider/FTPFileOperations",
 DownloadUrl =
"http://localhost:{port}/api/FTPProvider/FTPDownload",
 UploadUrl = "http://localhost:{port}/api/FTPProvider/FTPUUpload",
 GetImageUrl =
"http://localhost:{port}/api/FTPProvider/FTPGetImage"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

## FTP-FILE-SYSTEM.CS

**Note:** To learn more about the file actions that can be performed with File Transfer Protocol file system provider, refer to this [link](#).

### SQL database file system provider

In ASP.NET Core, SQL database file system provider allows the users to manage the file system being maintained in a SQL database table. Unlike the other file system providers, the SQL database file system provider works on ID basis. Here, each file and folder have a unique ID based on which all the file operations will be performed. To get started, clone the [sql-server-database-aspcore-file-provider](#) using the following command.

```
`typescript
<add name="FileExplorerConnection" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\FileManager.mdf;Integrated
Security=True;Trusted_Connection=true" />
`
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. To establish the SQL server connection with the database file (for eg: FileManager.mdf), specify the connection string in the web config file as follows.

```
`typescript
<add name="FileExplorerConnection" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\FileManager.mdf;Integrated
Security=True;Trusted_Connection=true" />
`
```

Then, make an entry for the connection string in `appsettings.json` file as follows.

```
`typescript
"ConnectionStrings": {
 "FileManagerConnection": "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\App_Data\\FileManager.mdf;In
tegrated Security=True;Connect Timeout=30"
}
`
```

Now, to configure the database connection, set the connection name, table name and root folder ID value by passing these values to the SetSQLConnection method.

```
`typescript
void SetSQLConnection(string name, string tableName, string tableID)
`
```

---

**Note:** Refer to this [FileManager.mdf](#), to learn about the pre-defined file system SQL database for the EJ2 File Manager.

---

After configuring the connection, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the files in the SQL database table.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with ASP.NET Core service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url =
"http://localhost:{port}/api/SQLProvider/SQLFileOperations",
 GetImageUrl =
"http://localhost:{port}/api/SQLProvider/SQLGetImage",
 UploadUrl = "http://localhost:{port}/api/SQLProvider/SQLUpload",
 DownloadUrl =
"http://localhost:{port}/api/SQLProvider/SQLDownload"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### SQL-FILE-SYSTEM.CS

---

**Note:** To learn more about file actions that can be performed with SQL database file system provider, refer to this [link](#).

---

#### NodeJS file system provider

In ASP.NET Core, NodeJS file system provider allows the users to manage the files and folders in a physical file system. It provides methods for performing all basic file operations like creating a folder, copy, move, delete, and download files and folders in the file system. We can use of the NodeJS file system provider either by installing the [ej2-filemanager-node-filessystem](#) package or by cloning the [file system provider](#) from the GitHub.

#### Using ej2-filemanager-node-filessystem package

- Install the ej2-filemanager-node-filessystem package by running the below command.

```
`typescript
```

```
npm install @syncfusion/ej2-filemanager-node-filessystem
```

```
`
```



- After installing the package, navigate to the ej2-filemanager-node-filesystem package folder within the node-modules.
- Run the command **npm install** command.

#### Cloning the ej2-filemanager-node-filesystem from GitHub

- Clone the ej2-filemanager-node-filesystem using the following command.

```
`typescript
```

```
git clone https://github.com/SyncfusionExamples/ej2-filemanager-node-filesystem.git node-filesystem-provider
```

```
`
```

- After cloning, open the root folder and run the command **npm install** command.

After installing the packages, set the root folder directory of the physical file system in the package JSON under scripts sections as follows.

```
`typescript
```

```
"start": "node filesystem-server.js -d D:/Projects"
```

```
`
```

---

**Note:** By default, the root directory will be configured to set C:/Users as the root directory.

---

To set the port in which the project to be hosted and the root directory of the file system. Run the following command.

```
`typescript
```

```
set PORT=3000 && node filesystem-server.js -d D:/Projects
```

```
`
```

---

**Note:** By default, the service will run 8090 port.

---

Now, just mapping the **ajaxSettings** property of the FileManager component to the appropriate file operation methods in the filesystem-server.js file will allow to manage the physical file system with NodeJS file system provider.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with NodeJS service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "http://localhost:{port}",
 GetImageUrl = "http://localhost:{port}/GetImage",
 UploadUrl = "http://localhost:{port}/Upload",
 DownloadUrl = "http://localhost:{port}/Download"
 }).Render()
```

```
<!-- end of filemanager element -->
</div>
</div>
```

## NODE-JS.CS

**Note:** To learn more about file actions that can be performed with NodeJS file system provider, refer to this [link](#).

### Google Drive file system provider

In ASP.NET Core, Google Drive file system provider allows the users to manage the files and folders in a Google Drive account. The Google Drive file system provider works on ID basis where each file and folder have a unique ID. To get started, clone the [google-drive-aspcore-file-provider](#) using the following command.

```
`typescript
git clone https://github.com/SyncfusionExamples/google-drive-aspcore-file-provider google-drive-aspcore-file-provider
cd google-drive-aspcore-file-provider
`
```

Google Drive file system provider use the [Google Drive APIs](#) to read the file in the file system and uses the [OAuth 2.0](#) protocol for authentication and authorization. To authenticate from the client end, obtain OAuth 2.0 client credentials from the [Google API Console](#). To learn more about generating the client credentials from the from Google API Console, refer to this [link](#).

After generating the client secret data, copy the JSON data to the following specified JSON files in the cloned location.

- EJ2GoogleDriveFileProvider > credentials > client\_secret.json
- GoogleOAuth2.0Base > credentials > client\_secret.json

After updating the credentials, just build and run the project. Now, the project will be hosted in `http://localhost:{port}`, and it will ask to log on to the Gmail account created the client secret credentials. Then, provide permission to access the Google Drive files by clicking the allow access button in the page. Now, just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods will allows to manage the files from the Google Drive.

## CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with ASP.NET Core service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
```

```
 Url =
 "http://localhost:{port}/api/GoogleDriveProvider/GoogleDriveFileOperations",
 GetImageUrl =
 "http://localhost:{port}/api/GoogleDriveProvider/GoogleDriveGetImage",
 UploadUrl =
 "http://localhost:{port}/api/GoogleDriveProvider/GoogleDriveUpload",
 DownloadUrl =
 "http://localhost:{port}/api/GoogleDriveProvider/GoogleDriveDownload"
 }).Render()
 <!-- end of filemanager element -->
</div>
</div>
```

### GOOGLE-DRIVE-FILE-SYSTEM.CS

**Note:** To learn more about file actions that can be performed with Google Drive file system provider, refer to this [link](#).

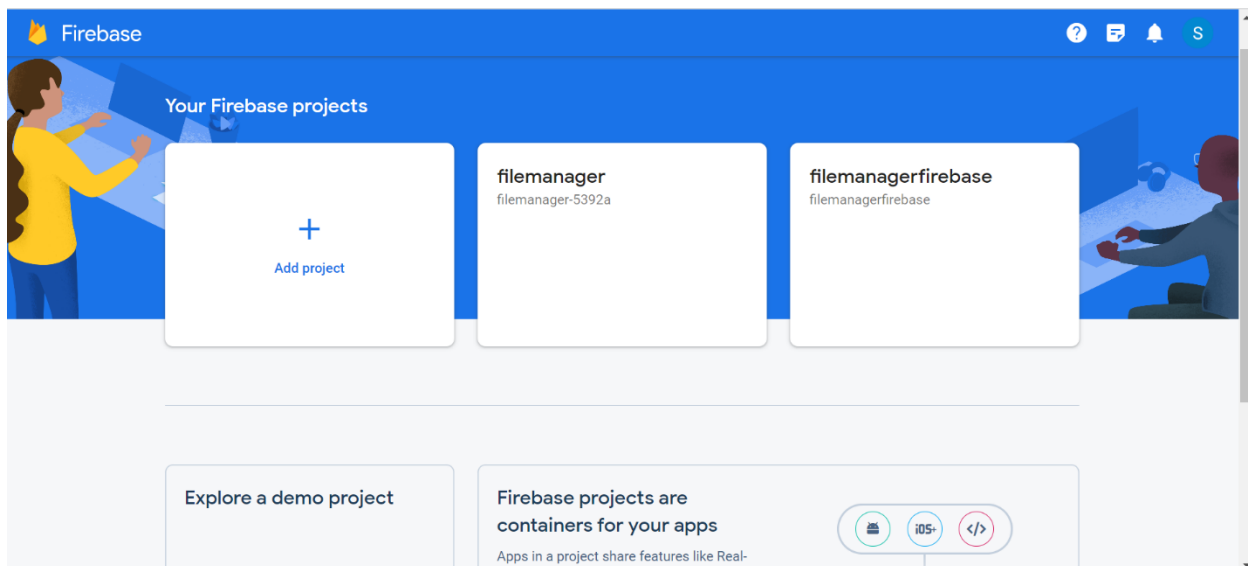
#### Firebase Realtime Database file system provider

The [Firebase Realtime Database](#) file system provider in **ASP.NET Core** provides the efficient way to store the File Manager file system in a cloud database as JSON representation.

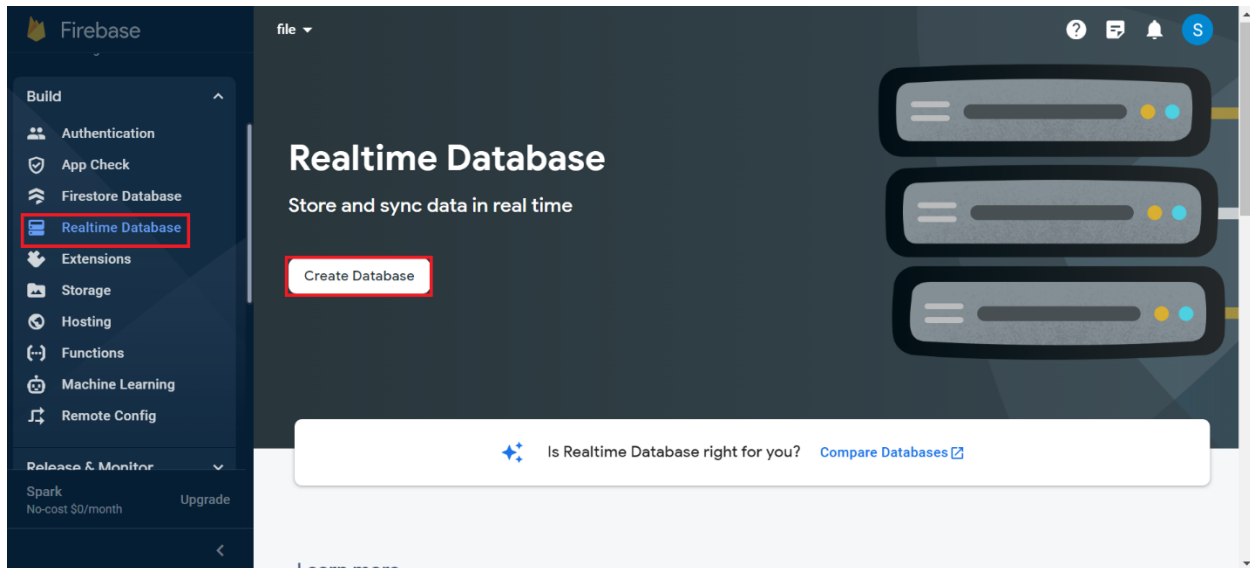
#### *Generate Secret access key from service account*

Follow the given steps to generate the secret access key:

- To access the Firebase console, please click on this [link](#). Once you have accessed the console, you can create a new project by filling in the necessary fields and clicking on the relevant buttons.



- Within the Firebase console, navigate to the **Build** tab. Under this tab, select the option for **Realtime Database**. From there, you can create a new database by clicking on the **Create Database** button.



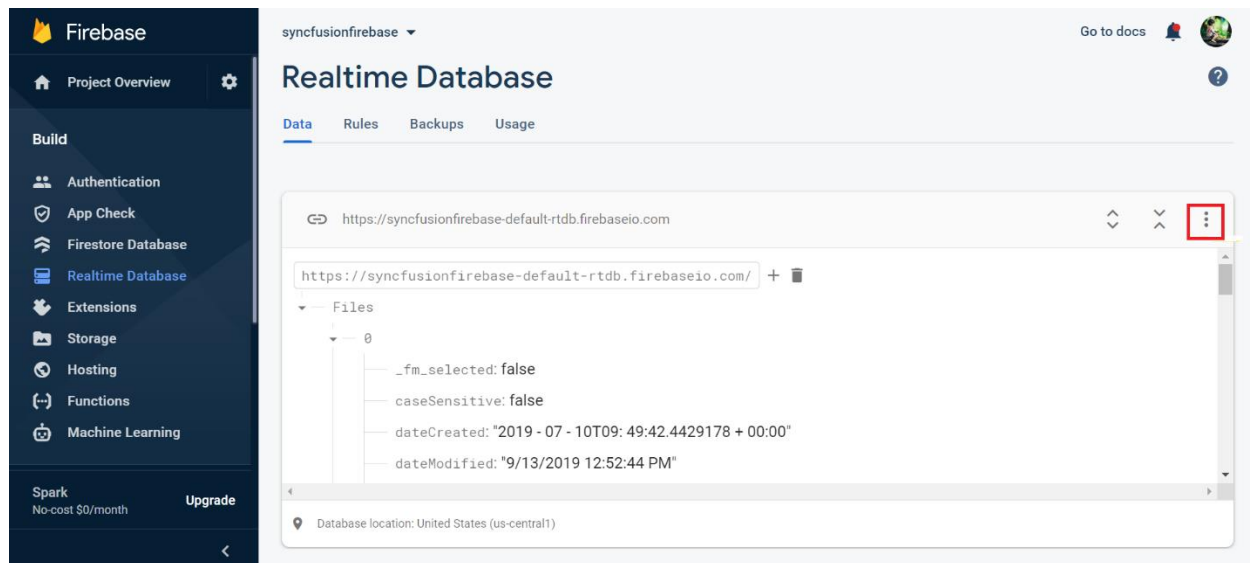
- To get started, create a root node and add any desired children to it. Please refer to the following code snippet for guidance on the structure of the JSON:

```
`typescript
{
 "Files" : [{
 "caseSensitive" : false,
 "dateCreated" : "8/22/2019 5:17:55 PM",
 "dateModified" : "8/22/2019 5:17:55 PM",
 "filterId" : "0/",
 "filterPath" : "/",
 "hasChild" : false,
 "id" : "5",
 "isFile" : false,
 "isRoot" : true,
 "name" : "Music",
 "parentId" : "0",
 "selected" : false,
 "showHiddenItems" : false,
 "size" : 0,
```

```
"type" : "folder"
},
{
 "caseSensitive" : false,
 "dateCreated" : "8/22/2019 5:18:03 PM",
 "dateModified" : "8/22/2019 5:18:03 PM",
 "filterId" : "0/",
 "filterPath" : "/",
 "hasChild" : false,
 "id" : "6",
 "isFile" : false,
 "isRoot" : true,
 "name" : "videos",
 "parentId" : "0",
 "selected" : false,
 "showHiddenItems" : false,
 "size" : 0,
 "type" : ""
}]
}
```

Here, the `Files` denotes the `rootNode` and the subsequent object refers to the children of the root node. `rootNode` will be taken as the root folder of the file system loaded which will be loaded in File Manager component.

- To import a JSON file into the Firebase Realtime Database, navigate to the **Data** tab and click on the action icon shown in the accompanying image. From there, select the **Import JSON** option and upload the JSON file that was created using the code provided above.



- To interact with the Firebase Realtime Database through your application, it is necessary to grant read and write permissions by defining appropriate rules in the Firebase project's **Rules tab**, as shown in the following code snippet. Once you have specified the rules, you can publish them by clicking the **Publish** button to enable the necessary authentication.

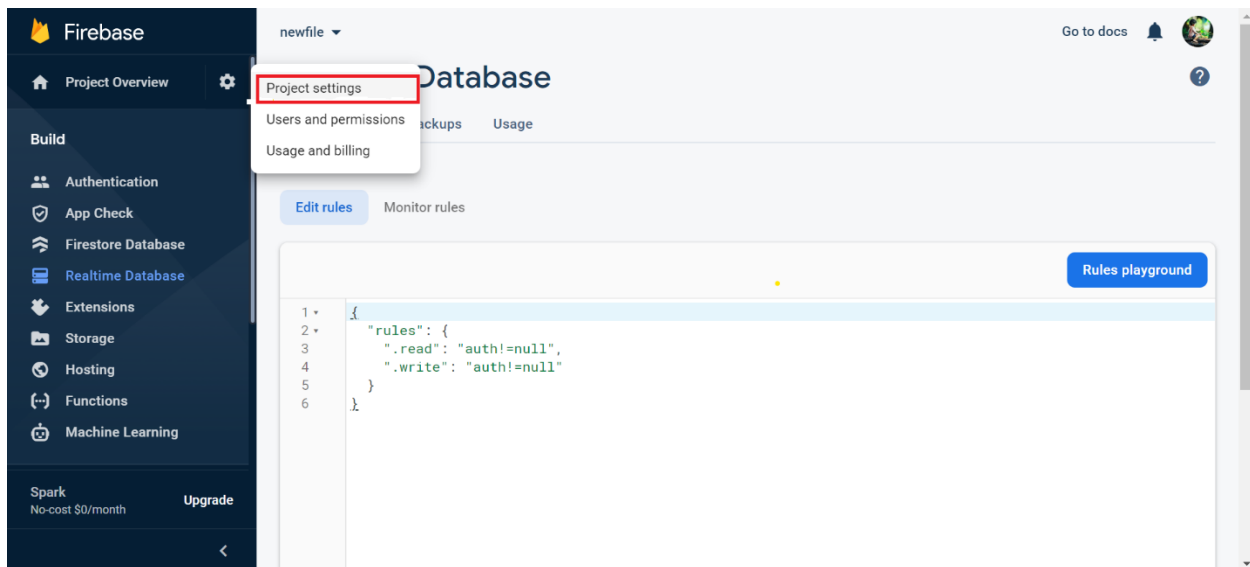
```
`typescript
{
/ Visit https://firebase.google.com/docs/database/security to learn more about security rules. /
"rules": {
".read": "auth!=null",
".write": "auth!=null"
}
}
`
```

---

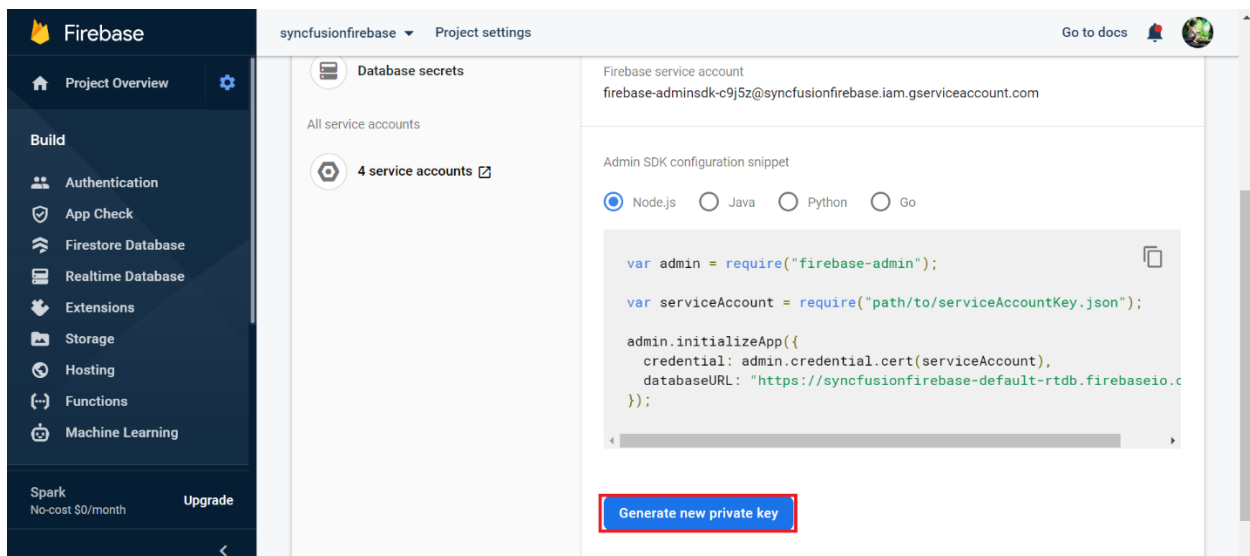
**Note:** By default, rules of a Firebase project will be **false**. To read and write the data, configure the **Rules** as given in the following code snippet in the *Rules* tab in the Firebase Realtime Database project.

---

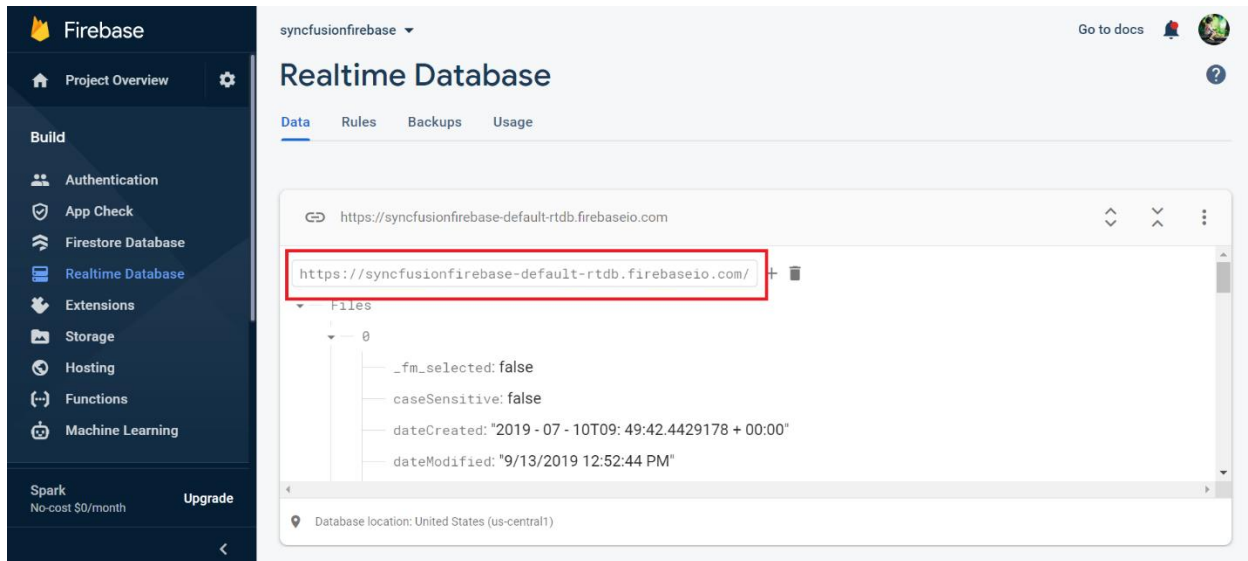
- Navigate to the project settings as instructed and then click on the **Service Account** tab.



- To obtain the access key JSON file, simply click on the **Generate new private key** button and then confirm by clicking the **Generate key** button in the pop-up window that appears.



- Next, you will need to clone the [firebase-realtime-database-apscore-file-provider](#) repository. Once cloned, simply open the project in Visual Studio and restore the NuGet package.
- Once you have generated the secret key, you will need to replace the JSON in the `access_key.json` file in the Firebase Realtime Database provider project with the newly generated key. This will enable authentication and allow you to perform read and write operations.
- In the **Data** tab, locate the project API URL and then paste it into the below mentioned section.



Register the Firebase Realtime Database by assigning *Firebase Realtime Database REST API link*, *rootNode*, and *serviceAccountKeyPath* parameters in the `RegisterFirebaseRealtimeDB` method of class `FirebaseRealtimeDBFileProvider` in the controller part of the ASP.NET Core application.

```
`typescript
```

```
this.operation.RegisterFirebaseRealtimeDB(string apiUrl, string rootNode, string serviceAccountKeyPath)
```

```
,
```

**Example:**

```
`typescript
```

```
this.operation.RegisterFirebaseRealtimeDB("{copy your API URL here}", "Files",
hostingEnvironment.ContentRootPath + "\\FirebaseRealtimeDBHelper\\access_key.json");
```

```
,
```

In the above code,

- `{copy your API URL here}` denotes Firebase Realtime Database REST API link.
- `Files` denotes newly created root node in Firebase Realtime Database.
- `hostingEnvironment.ContentRootPath + "\\FirebaseRealtimeDBHelper\\access_key.json` denotes service account key path which has authentication key for the Firebase Realtime Database data.

After configuring the Firebase Realtime Database service link, build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the `ajaxSettings` property of the File Manager component to the appropriate controller methods allows to manage the files in the Firebase Realtime Database.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with Firebase service-->
```



```
<!--Replace the hosted port number in the place of "{port}"-->
@Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
{
 Url =
"http://localhost:{port}/api/FirebaseProvider/FirebaseRealtimeFileOperations
",
 GetImageUrl =
"http://localhost:{port}/api/FirebaseProvider/FirebaseRealtimeGetImage",
 UploadUrl =
"http://localhost:{port}/api/FirebaseProvider/FirebaseRealtimeUpload",
 DownloadUrl =
"http://localhost:{port}/api/FirebaseProvider/FirebaseRealtimeDownload"
}).Render()
<!-- end of filemanager element -->
</div>
</div>
```

### **FIRE-BASE.CS**

**Note:** To learn more about the file actions that can be performed with Firebase Realtime Database file system provider, refer to this [link](#).

#### IBM Cloud Object Storage file provider

The IBM Cloud Object Storage file provider module allows you work with the IBM Cloud Object Storage. It also provides the methods for performing various file actions such as creating a new folder, renaming files, and deleting files. The IBM Cloud Object Storage file provider serves the file provider support for the File Manager component with the IBM Cloud Object Storage. We can make use of IBM Cloud Object Storage file provider by installing the [ej2-filemanager-ibm-cos-node-file-provider](#) npm package or by cloning the [file provider](#) from the GitHub.

#### *Using ej2-filemanager-ibm-cos-node-file-provider npm package*

- Install the ej2-filemanager-ibm-cos-node-file-provider npm package by running the below command.

```
`typescript
```

```
npm install @syncfusion/ej2-filemanager-ibm-cos-node-file-provider
```

```
,
```

- After installing the package, navigate to the ej2-filemanager-ibm-cos-node-file-provider package folder within the node-modules.
- Run the **npm install** command to install the dependent packages for file provider.

#### *Cloning the filemanager-ibm-cos-node-file-provider from GitHub*

- Clone the filemanager-ibm-cos-node-file-provider using the following command.

```
`typescript
```

```
git clone https://github.com/SyncfusionExamples/filemanager-ibm-cos-node-file-provider.git
```

```
,
```

- After cloning, open the root folder and run the command **npm install** command.

To set the port in which the project to be hosted. Run the following command.

```
`typescript
```

```
set PORT=3000 && node index.js
```

```
,
```

---

**Note:** By default, the service will run 8090 port.

---

Now, just mapping the **ajaxSettings** property of the FileManager component to the appropriate file operation methods in the index.js file will allow to manage the IBM Cloud Object Storage.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Initialize file manager component with IBM COS service-->
 <!--Replace the hosted port number in the place of "{port}"-->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "http://localhost:{port}",
 GetImageUrl = "http://localhost:{port}/GetImage",
 UploadUrl = "http://localhost:{port}/Upload",
 DownloadUrl = "http://localhost:{port}/Download"
 }).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

### IBM-JS.CS

---

**Note:** To learn more about the file actions that can be performed with IBM Cloud Object Storage file provider, refer to this [link](#).

---

### Localization

The file manager can be localized to any culture by defining the texts and messages of the file manager in the corresponding culture. The default locale of the file manager is **en**(English). The following table represents the default texts and messages of the file manager in **en** culture.

KEY	Text/Message
-----	--------------

----	----
------	------

NewFolder	New folder
-----------	------------

Upload	Upload
Delete	Delete
Rename	Rename
Download	Download
Cut	Cut
Copy	Copy
Paste	Paste
SortBy	Sort by
Refresh	Refresh
Item-Selection	item selected
Items-Selection	items selected
View	View
Details	Details
SelectAll	Select all
Open	Open
Tooltip-NewFolder	New folder
Tooltip-Upload	Upload
Tooltip-Delete	Delete
Tooltip-Rename	Rename
Tooltip-Download	Download
Tooltip-Cut	Cut
Tooltip-Copy	Copy
Tooltip-Paste	Paste
Tooltip-SortBy	Sort by
Tooltip-Refresh	Refresh
Tooltip-Selection	Clear selection
Tooltip-View	View
Tooltip-Details	Details
Tooltip-SelectAll	Select all
Name	Name
Size	Size
DateModified	Modified
DateCreated	Date created

Path	Path
Created	Created
Modified	Modified
Location	Location
Type	Type
Permission	Permission
Ascending	Ascending
Descending	Descending
None	None
View-LargeIcons	Large icons
View-Details	Details
Search	Search
Button-Ok	OK
Button-Cancel	Cancel
Button-Yes	Yes
Button-No	No
Button-Create	Create
Button-Save	Save
Header-NewFolder	Folder
Content-NewFolder	Enter your folder name
Header-Rename	Rename
Content-Rename	Enter your new name
Header-Rename-Confirmation	Rename Confirmation
Content-Rename-Confirmation	If you change a file name extension
Header-Delete	Delete File
Content-Delete	Are you sure you want to delete this file?
Header-Multiple-Delete	Delete Multiple Files
Content-Multiple-Delete	Are you sure you want to delete these {0} files?
Header-Folder-Delete	Delete Folder
Content-Folder-Delete	Are you sure you want to delete this folder?
Header-Duplicate	File exists
Content-Duplicate	already exists. Are you sure you want to replace it?
Header-Upload	Upload Files

|Error|Error|

|Validation-Empty|The file or folder name cannot be empty.|

|Validation-Invalid|The file or folder name {0} contains invalid characters. Use a different name. Valid file or folder names cannot end with a dot or space, and cannot contain any of the following characters: \\/:\*?"<>\\||

|Validation-NewFolder-Exists|A file or folder with the name {0} already exists.|

|Validation-Rename-Exists|Cannot rename {0} to {1}|

|Folder-Empty|This folder is empty|

|File-Upload|Drag files here to upload|

|Search-Empty|No results found|

|Search-Key|Try with different keywords|

|Filter-Empty|No results found|

|Filter-Key|Try with different filter|

|Sub-Folder-Error|The destination folder is the subfolder of the source folder|

|Same-Folder-Error|The destination folder is the same as the source folder.|

|Access-Denied|Access Denied|

|Access-Details|You don't have permission to access this folder|

|Header-Retry|File Already Exists|

|Content-Retry|A file with this name already exists in this folder. What would you like to do?|

|Button-Keep-Both|Keep both|

|Button-Replace|Replace|

|Button-Skip|Skip|

|ApplyAll-Label|Do this for all current items|

|KB|KB|

|Access-Message|{0} is not accessible. You need permission to perform the {1} action.|

|Network-Error|NetworkError: Failed to send on XMLHttpRequest: Failed to load|

|Server-Error|ServerError: Invalid response from|

The below example shows adding the German culture locale(**de-DE**)

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings()
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
```

```

 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).Locale("de").Render()
 <!-- end of filemanager element -->
</div>
</div>
<script>
 // Declare locale value
 ej.base.L10n.load({
 'de': {
 'filemanager': {
 "NewFolder": "Neuer Ordner",
 "Upload": "Hochladen",
 "Delete": "Löschen",
 "Rename": "Umbenennen",
 "Download": "Herunterladen",
 "Cut": "Schnitt",
 "Copy": "Kopieren",
 "Paste": "Einfügen",
 "SortBy": "Sortiere nach",
 "Refresh": "Aktualisierung",
 "Item-Selection": "Artikel ausgewählt",
 "Items-Selection": "Elemente ausgewählt",
 "View": "Aussicht",
 "Details": "Einzelheiten",
 "SelectAll": "Wählen Sie Alle",
 "Open": "Öffnen",
 "Tooltip-NewFolder": "Neuer Ordner",
 "Tooltip-Upload": "Hochladen",
 "Tooltip-Delete": "Löschen",
 "Tooltip-Rename": "Umbenennen",
 "Tooltip-Download": "Herunterladen",
 "Tooltip-Cut": "Schnitt",
 "Tooltip-Copy": "Kopieren",
 "Tooltip-Paste": "Einfügen",
 "Tooltip-SortBy": "Sortiere nach",
 "Tooltip-Refresh": "Aktualisierung",
 "Tooltip-Selection": "Auswahl aufheben",
 "Tooltip-View": "Aussicht",
 "Tooltip-Details": "Einzelheiten",
 "Tooltip-SelectAll": "Wählen Sie Alle",
 "Name": "Name",
 "Size": "Größe",
 "DateModified": "Geändert",
 "DateCreated": "Datum erstellt",
 "Path": "Pfad",
 "Modified": "Geändert",
 "Created": "Erstellt",
 "Location": "Ort",
 "Type": "Art",
 "Permission": "Genehmigung",
 "Ascending": "Aufsteigend",
 "Descending": "Absteigend",
 "None": "Keiner",
 "View-LargeIcons": "Große Icons",
 "View-Details": "Einzelheiten",
 "Search": "Suche",
 }
 }
 });

```

```

 "Button-Ok": "OK",
 "Button-Cancel": "Stornieren",
 "Button-Yes": "Ja",
 "Button-No": "Nein",
 "Button-Create": "Erstellen",
 "Button-Save": "Sparen",
 "Header-NewFolder": "Mappe",
 "Content-NewFolder": "Geben Sie Ihren Ordnernamen ein",
 "Header-Rename": "Umbenennen",
 "Content-Rename": "Geben Sie Ihren neuen Namen ein",
 "Header-Rename-Confirmation": "Bestätigung umbenennen",
 "Content-Rename-Confirmation": "Wenn Sie eine
Dateinamenerweiterung ändern, wird die Datei möglicherweise instabil.
Möchten Sie sie wirklich ändern?",
 "Header-Delete": "Datei löschen",
 "Content-Delete": "Möchten Sie diese Datei wirklich
löschen?",
 "Header-Multiple-Delete": "Mehrere Dateien löschen",
 "Content-Multiple-Delete": "Möchten Sie diese {0}
Dateien wirklich löschen?",
 "Header-Folder-Delete": "Lösche Ordner",
 "Content-Folder-Delete": "Möchten Sie diesen Ordner
wirklich löschen?",
 "Header-Duplicate": "Datei / Ordner existiert",
 "Content-Duplicate": "{0} existiert bereits. Möchten Sie
umbenennen und einfügen?",
 "Header-Upload": "Daten hochladen",
 "Error": "Error",
 "Validation-Empty": "Der Datei - oder Ordnername darf
nicht leer sein.",
 "Validation-Invalid": "Der Datei- oder Ordnername {0}
enthält ungültige Zeichen. Bitte verwenden Sie einen anderen Namen. Gültige
Datei- oder Ordnernamen dürfen nicht mit einem Punkt oder Leerzeichen enden
und keines der folgenden Zeichen enthalten: \\ /: *? \" < > | ",
 "Validation-NewFolder-Exists": "Eine Datei oder ein
Ordner mit dem Namen {0} existiert bereits.",
 "Validation-Rename-Exists": "{0} kann nicht in {1}
umbenannt werden: Ziel existiert bereits.",
 "Folder-Empty": "Dieser Ordner ist leer",
 "File-Upload": "Dateien zum Hochladen hierher ziehen",
 "Search-Empty": "Keine Ergebnisse gefunden",
 "Search-Key": "Versuchen Sie es mit anderen
Stichwörtern",
 "Filter-Empty": "keine Ergebnisse gefunden",
 "Filter-Key" : "Versuchen Sie es mit einem anderen
Filter",
 "Sub-Folder-Error": "Der Zielordner ist der Unterordner
des Quellordners.",
 "Same-Folder-Error": "Der Zielordner ist derselbe wie
der Quellordner.",
 "Access-Denied": "Zugriff verweigert",
 "Access-Details": "Sie haben keine Berechtigung, auf
diesen Ordner zuzugreifen.",
 "Header-Retry": "Die Datei existiert bereits",
 "Content-Retry": "In diesem Ordner ist bereits eine
Datei mit diesem Namen vorhanden. Was möchten Sie tun?",
 "Button-Keep-Both": "Behalte beides",

```

```

 "Button-Replace": "Ersetzen",
 "Button-Skip": "Überspringen",
 "ApplyAll-Label": "Mache das für alle aktuellen
Artikel",
 "KB": "KB",
 "Access-Message": "{0} ist nicht zugänglich. Sie
benötigen die Berechtigung, um die Aktion {1} auszuführen.",
 "Network-Error": "NetworkError: Fehler beim Senden auf
XMLHttpRequest: Fehler beim Laden",
 "Server-Error": "ServerError: Ungültige Antwort von"
 }
 });
</script>

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 }
 }
 }
}

```



```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 JsonResult(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 JsonResult(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 JsonResult(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)

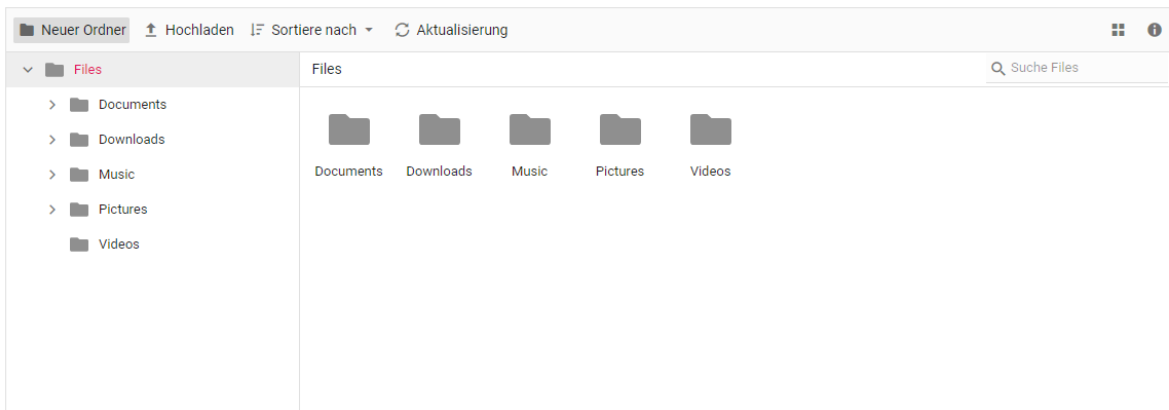
```

```

 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
 }
 // Processing the Download operation
 public ActionResult Download(string downloadInput)
 {
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
 }
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.



## Virtualization in File Manager Component

File Manager's UI virtualization allows you for the dynamic loading of a large number of directories and files in both the detailsView and largeIconsView without degrading its performance.

### Enable Virtualization

The virtualization of the File Manager component is based on the height and width of the viewport. The items will be loaded in both [largeIconsView](#) and [detailsView](#) based on the viewport size.

In order to enable [virtualization](#), you must set the [EnableVirtualization](#) property to true.

In the instance below, a sizable collection of files can be found in the folders **Documents** and **Text Documents**.

### CSHTML

```
<div class="control-section">
<div class="sample-container">
 <!-- File Manager virtualization feature sample -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(
 new Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/FileOperations",
 DownloadUrl = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/Download",
 UploadUrl = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/Upload",
 GetImageUrl = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/GetImage"
 }).EnableVirtualization(true).Render()
</div>
</div>
```

### HOMECONTROLLER\_MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
```

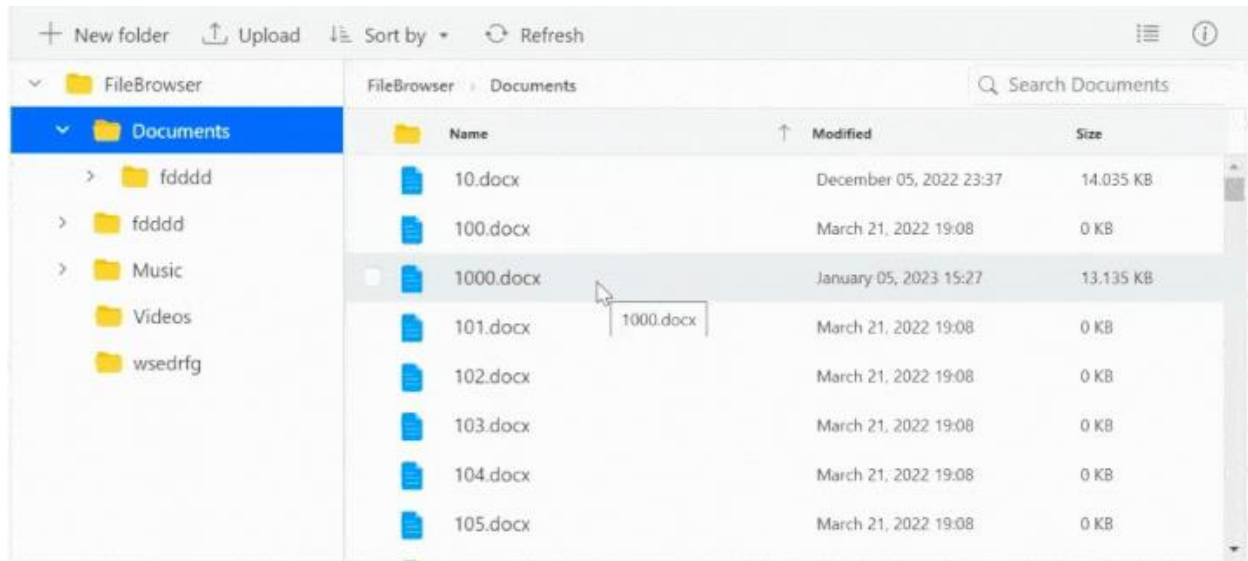
```

 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation

```

```
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
```

Output be like the below.



### Limitations for Virtualization

- Programmatic selection using the **selectAll** method is not supported with virtual scrolling.
- The keyboard shortcut **CTRL+A** will only select the files and directories that are currently visible within the viewport, rather than selecting all files and directories in the entire directory tree.
- Selected file items are not maintained while scrolling, considering the performance of the component.

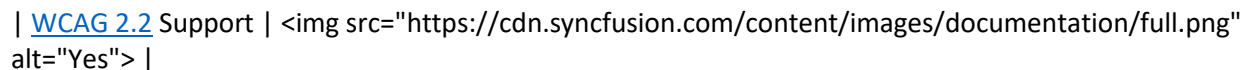
### Accessibility in ASP.NET MVC File Manager component

The File Manager component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the File Manager component is outlined below.

| Accessibility Criteria | Compatibility |

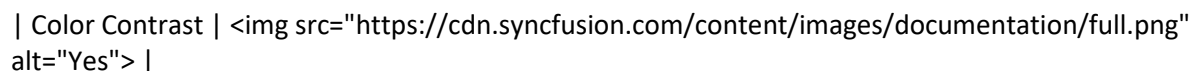
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |

```
| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

#### WAI-ARIA attributes

The File Manager component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the File Manager component:

Attributes	Purpose
role	Used to convey a significant and contextual message to the user.
aria-disabled	Indicates whether the File Manager component is in disabled state.
aria-haspopup	Indicates whether the toolbar item has a popup list or not.
aria-orientation	Indicates whether the File Manager element is oriented horizontally or vertically.
aria-expanded	Indicates whether the Treeview node has been expanded.
aria-owns	Contains the ID of the suggestion list to indicate popup as a child element.
aria-activedescendent	Holds the ID of the active list item to focus its descendant child element.
aria-level	Specifies the level of the element in Treeview Structure.
aria-selected	Indicates whether a particular node is in selected state.
aria-placeholder	Represents a hint (word or phrase) to the user about what to enter in the text field.
aria-label	Provides an accessible name for the element.

- | **aria-checked** | Indicates whether the checkbox is in checked state. |
- | **aria-labelledby** | Provides a label for the dialog. Typically, the "aria-labelledby" attribute will contain the id of the element used as the dialog's title. |
- | **aria-describedby** | This attribute points to the Dialog element describing the one it's set on. |
- | **aria-modal** | Indicates whether an element is a modal when display. |
- | **aria-colcount** | Specifies the number of columns in full table. |
- | **aria-colindexnt** | Defines the number of columns within a table in details view. |
- | **aria-rowspan** | Defines the number of rows a cell spanned within a table in details view. |
- | **aria-colspan** | Defines the number of columns a cell spanned within a table in details view. |
- | **aria-sort** | Indicates whether items in the table are sorted in ascending or descending order. |
- | **aria-grabbed** | When the folder/file item is chosen for dragging, the aria-grabbed attribute is set to "true." If it's set to "false," the element can be grabbed for drag-and-drop, but it won't be actively held. |
- | **aria-busy** | This attribute is set to false when table content is loaded. |
- | **aria-multiselectable** | Defines more than one item has been selected. |

#### Keyboard interaction

The File Manager component followed the **keyboard interaction** guidelines, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the File Manager component.

| **Press** | **To do this** |

| --- | --- |

| **Page Down** | Scrolls down to the next folder or file and selects the first item when files are loaded. |

| **Page Up** | Scrolls up to previous folder and select the first item when files are loaded. |

| **Enter** | Selects the focused item and navigate through the child elements. |

| **Tab** | Focuses on the first element of toolbar and navigates through the next tab indexed element. |

| **Esc(Escape)** | Closes the image when it is in open state. |

| **Alt+N** | Creates a new folder dialog. |

| **F5** | Refresh the file manager element. |

| **Home** | Navigate through the first element of details view or large icons view. |

| **End** | Navigate through the last element of details view or large icons view. |

| **Move Left** | Scrolls left to the previous folder and select the first item when files are loaded |



Move Right	Scrolls right to the previous folder and select the first item when files are loaded	
Alt+Enter	Shows the get details info for selected folder.	
Shift+Right	Allows multiselection. Select the file or folder at the right of the previously selected folder.	
Shift+Left	Allows multiselection. Select the file or folder at the left of the previously selected folder.	
Shift+Down	Allows multiselection. Select the file or folder till the focused index.	
Shift+Delete	Permanently deletes the selected file or folder in the file manager element.	
Delete	Deletes the selected file or folder in the file manager element.	
Shift+Up	Allows multiselection. Select the file or folder till the focused index.	
Ctrl+C	Copies the selected file or folder in the file manager element.	
Ctrl+V	Pastes the copied/cut file or folder in the file manager element.	
Ctrl+X	Cuts the selected file or folder in the file manager element.	
Ctrl+A	Select all the files or folders in the details view or large icons view.	
F2	Creates a rename dialog for a selected file or folder in the file manager element.	
Shift+F10	Opens the context menu for the selected file or folder in the file manager element.	
Ctrl+D	Downloads the list of selected files or folders in the file manager element.	
Ctrl+Shift+1	Changes the file manager layout to details view.	
Ctrl+Shift+2	Changes the file manager layout to details view.	

### Ensuring accessibility

The File Manager component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the File Manager component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the File Manager component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Access Control in File Manager component

The FileManager allows you to define access permissions for folders and files using a set of access rules to user(s).

- [Access Rules](#)
- [Permissions](#)

### Access Rules

The FileAccessController allows you to define security permissions for folders and files using a set of folder or file access rules.

To set up access rules for folders (including their files and sub-folders) and individual files, use the SetRules() method. The following table represents the AccessRule properties available for file and folder:

Properties	Applicable for file	Applicable for folder	Description
Copy	Yes	Yes	Allows access to copy a file or folder.
Read	Yes	Yes	Allows access to read a file or folder.
Write	Yes	Yes	Allows permission to write a file or folder.
WriteContents	No	Yes	Allows permission to write the content of folder.
Download	Yes	Yes	Allows permission to download a file or folder.
Upload	No	Yes	Allows permission to upload to the folder.
Path	Yes	Yes	Specifies the path to apply the rules, which are defined.
Role	Yes	Yes	Specifies the role to which the rule is applied.
IsFile	Yes	Yes	Specifies whether the rule is specified for folder or file.

The following syntax represents the access Rules for Administrator using file or folder.

```
`typescript
//Administrator
//Access Rules for File
new AccessRule { Path = "/", Role = "Administrator", Read = Permission.Allow, Write =
Permission.Allow, Copy = Permission.Allow, Download = Permission.Allow, IsFile = true },
// Access Rules for folder
new AccessRule { Path = "*", Role = "Administrator", Read = Permission.Allow, Write = Permission.Allow,
Copy = Permission.Allow, WriteContents = Permission.Allow, Upload = Permission.Allow, Download =
Permission.Deny, IsFile = false },
`
```

The following syntax represent the access Rules for Default user using file or folder.

```
`typescript
//Default User
//Access Rules for File
new AccessRule { Path = "/", Role = "Default User", Read = Permission.Deny, Write = Permission.Deny,
Copy = Permission.Deny, Download = Permission.Deny, IsFile = true },
// Access Rules for folder
```

```
new AccessRule { Path = "*", Role = "Default User", Read = Permission.Deny, Write = Permission.Deny,
Copy = Permission.Deny, WriteContents = Permission.Deny, Upload = Permission.Deny, Download =
Permission.Deny, IsFile = false },
```

,

### Permissions

It helps to explain how to apply security permission to file manager file or folder using access rules. The following table represent the value that determines the permission.

Value	Description
-------	-------------

---	---
-----	-----

Allow	Allows you to do read, write, copy, and download operations.
-------	--------------------------------------------------------------

Deny	Denies you to do read, write, copy, and download operations.
------	--------------------------------------------------------------

Use the **Role** property to apply created roles to the file manager. After that, the file manager displays folder or file and allow permission based on assigned roles.

The following syntax represent how to apply permission based on assigned roles.

- Permission denied for administrator to write a file or folder.

```
`typescript
```

```
// For file
```

```
new AccessRule { Path = "/", Role = "Administrator", Read = Permission.Allow, Write = Permission.Deny,
IsFile = true},
```

```
// For folder
```

```
new AccessRule { Path = "*", Role = "Administrator", Read = Permission.Allow, Write = Permission.Deny,
IsFile = false},
```

,

The following syntax represent how to allow or deny permission based on file or folder access rule.

- Permission denied for writing except for particular file or folder.

```
`typescript
```

```
// Deny writing for particular folder
```

```
new AccessRule { Path = "/Documents", Role = "Document Manager", Read = Permission.Allow, Write =
Permission.Deny, Copy = Permission.Allow, WriteContents = Permission.Deny, Upload =
Permission.Deny, Download = Permission.Deny, IsFile = false },
```

```
// Deny writing for particular file
```

```
new AccessRule { Path = "/Pictures/Employees/Adam.png", Role = "Document Manager", Read =
Permission.Allow, Write = Permission.Deny, Copy = Permission.Deny, Download = Permission.Deny,
IsFile = true },
```

,

- Permission denied for writing and uploading in root folder.

```
` typescript
```

```
// Folder Rule
```

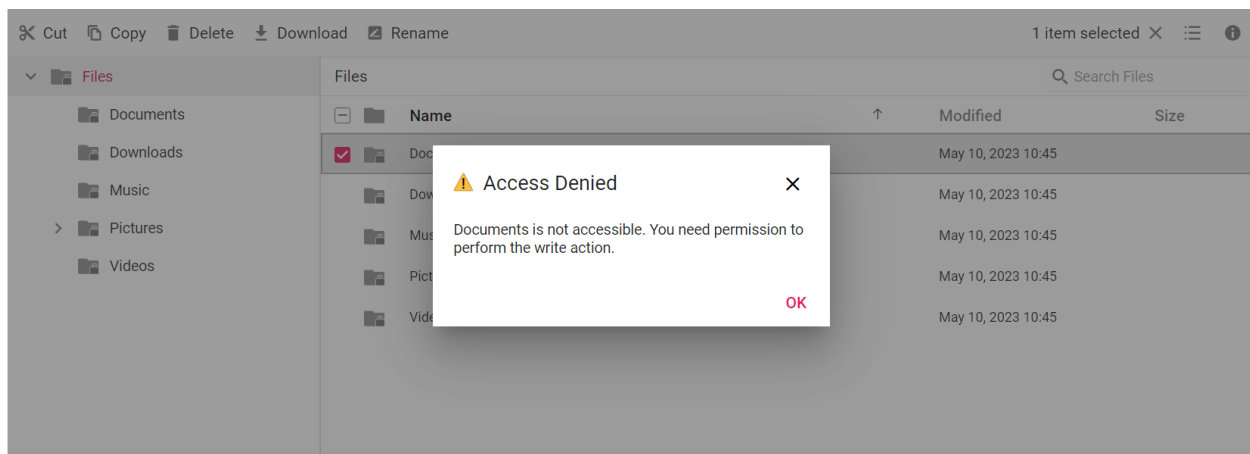
```
new AccessRule { Path = "/", Role = "Document Manager", Read = Permission.Allow, Write =
Permission.Deny, Copy = Permission.Deny, WriteContents = Permission.Deny, Upload =
Permission.Deny, Download = Permission.Deny, IsFile = false },
`
```

The following example demonstrate the file manager rendered with access control support.

### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManagerAccess/FileOperations",
 GetImageUrl = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManagerAccess/GetImage",
 UploadUrl = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManagerAccess/Upload",
 DownloadUrl = "https://ej2-aspcore-
service.azurewebsites.net/api/FileManagerAccess/Download"
 }).View(Syncfusion.EJ2.FileManager.ViewType.Details).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

Output be like the below, when write the documents folder.



### How To

How to add custom menu item in context menu

The context menu can be customized using the [contextMenuSettings](#), [menuOpen](#), and [menuClick](#) events.

The following example shows adding a custom item in the context menu.

The [menuOpen](#) event is used to add the new menu item. The [menuClick](#) event is used to add event handler to the new menu item.

### CSHTML

```
@{
 string[] files = new string[] { "Custom", "Open", "|", "Delete",
 "Download", "Rename", "|", "Details" };
 string[] folder = new string[] { "Custom", "Open", "|", "Delete",
 "Download", "Rename", "|", "Details" };
 string[] layout = new string[] { "Custom", "SortBy", "View", "Refresh",
 "|", "NewFolder", "Upload", "|", "Details", "|", "SelectAll" };
}
<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings()
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).ContextMenuSettings(new
 Syncfusion.EJ2.FileManager.FileManagerContextMenuSettings()
 {
 File = files,
 Folder = folder,
 Layout = layout
 }).MenuOpen("menuOpen").MenuClick("menuClick").Render()
 <!-- end of filemanager element -->
 </div>
</div>
<script>
// Icon added to custom menu item
function menuOpen(args) {
 for (var i = 0; i < args.items.length; i++) {
 if (args.items[i].id === this.element.id + '_cm_custom') {
 args.items[i].iconCss = 'e-icons e-fe-tick';
 }
 }
}
// event for custom menu item
function menuClick(args) {
 if (args.item.text === 'Custom') {
 alert('You have clicked custom menu item')
 }
}
</script>
```

### HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 }
 }
 }
}

```

```

 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()

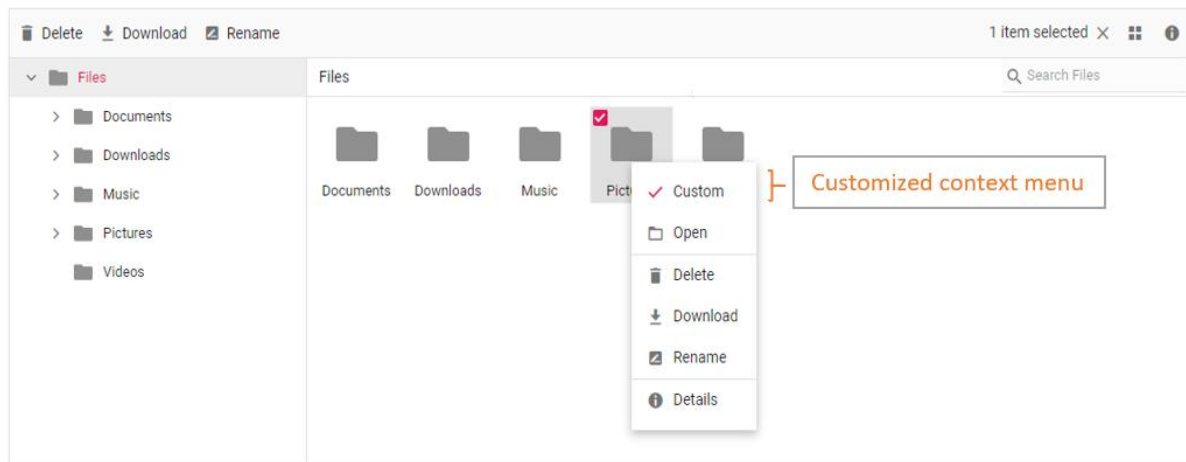
```

```

 {
 return View();
 }
}

```

Output be like the below.



### How to add custom button in toolbar

You can modify the items displayed in the toolbar by utilizing the [toolbarItems](#) API. To display both default and customized items, it's essential to assign a unique **name** to each item. Additionally, you have the flexibility to alter the default items by adjusting properties such as **tooltipText**, **iconCss**, **Text**, **suffixIcon** and more. This level of customization allows you to tailor the toolbar to your specific requirements and design preferences. The names used in the code example below serve as unique identifiers for default toolbar items, while custom items can be assigned any unique name value to distinguish them from the defaults.

For instance, here's an example of how to add a custom checkbox to the toolbar using the **template** property. Here we have modified the default **New Folder** item and added a custom toolbar item for selection.

### CSHTML

```

@{
 string[] items = new string[] { "NewFolder", "Upload", "Delete",
 "Download", "Rename", "SortBy", "Refresh", "Selection", "View", "Details",
 "Custom" };
}
<div class="control-section">
 <div class="sample-container">
 <!-- Filemanager element declaration -->
 @Html.EJS().FileManager("file").AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings()
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"

```



```

 }).ToolBarSettings(new
Syncfusion.EJ2.FileManager.FileManagerToolBarSettings()
 {
 Items = items
 }).ToolBarClick("toolbarClick").ToolBarCreate("onCreate").Render()
 <!-- end of filemanager element -->
</div>
</div>
<script>
 // event for custom toolbar item
 function toolbarClick(args) {
 if (args.item.text === 'Custom') {
 alert('You have clicked custom toolbar item')
 }
 }
 function onCreate(args) {
 for (var i = 0; i < args.items.length; i++) {
 if (args.items[i].id === this.element.id + '_tb_custom') {
 args.items[i].prefixIcon = 'e-icons e-fe-tick';
 }
 }
 }
</script>
<style>
 .e-fe-tick::before {
 content: '\e614';
 }
</style>

```

## HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 }
}

```

```

{
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 // created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 // SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 // which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 // file name; NewName - New file name
 }
}

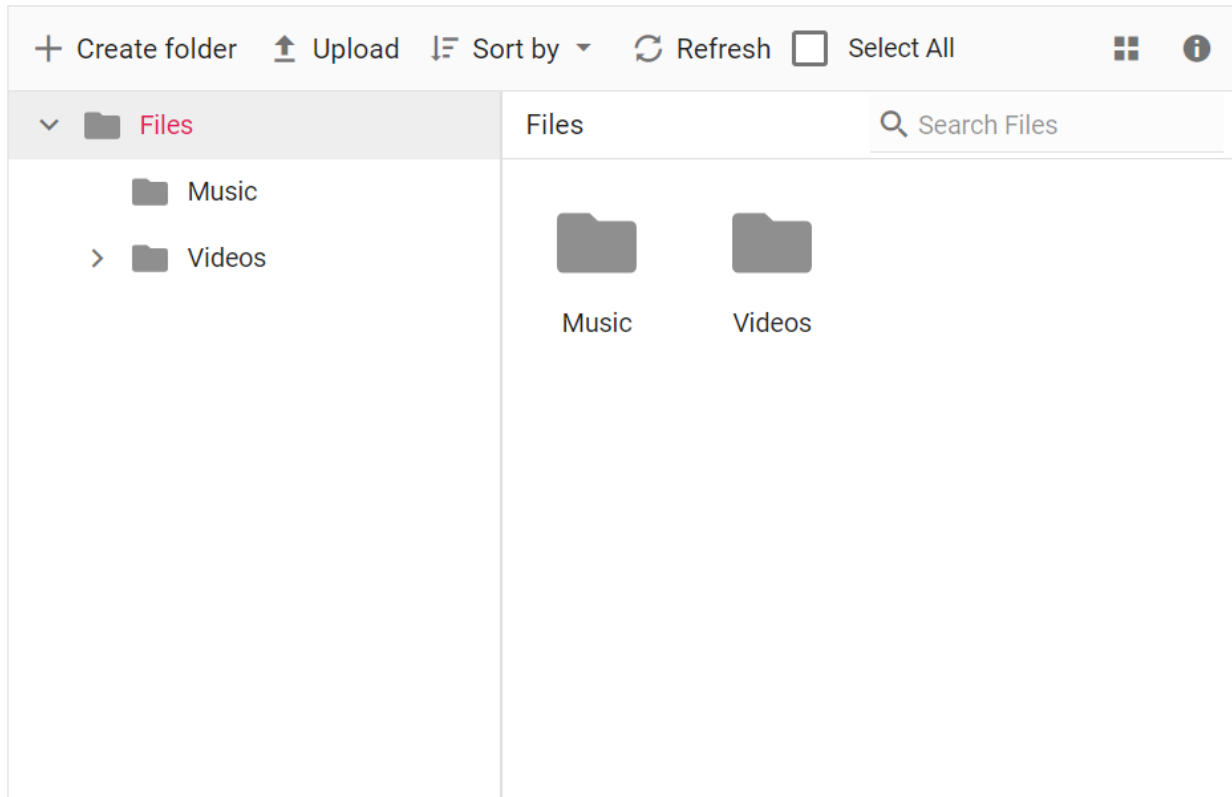
```

```

 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



### How to enable/disable toolbar item/items

The toolbar items can be enabled/disabled by specifying the items in `enableToolBarItems` or `disableToolBarItems` methods respectively.

The following example shows enabling and disabling toolbar items on button click.

### CSHTML

```
<div>
 <!-- Declare button element -->
 @Html.EJS().Button("enable").Content("Enable New Folder toolbar
item").CssClass("e-success").Render()
 @Html.EJS().Button("disable").Content("Disable New Folder toolbar
item").CssClass("e-danger").Render()
 <!-- end of button element -->
</div>
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).View(Syncfusion.EJ2.FileManager.ViewType.Details).Render()
 <!-- end of filemanager element -->
 </div>
</div>
```

```

</div>
<script>
 document.addEventListener('DOMContentLoaded', function () {
 var filemanagerInstance =
document.getElementById("filemanager").ej2_instances[0];
 // Click event for enable button
 document.getElementById("enable").onclick = function (args) {
 // Enable new folder toolbar item
 filemanagerInstance.enableToolbarItems(["newfolder"]);
 }
 // Click event for enable button
 document.getElementById("disable").onclick = function (args) {
 // Disable new folder toolbar item
 filemanagerInstance.disableToolbarItems(["newfolder"]);
 }
 });
</script>
<style>
 /*sample level styles*/
 #enable {
 margin: 10px;
 }
 #disable {
 margin: 10px;
 }
</style>

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations

```

```

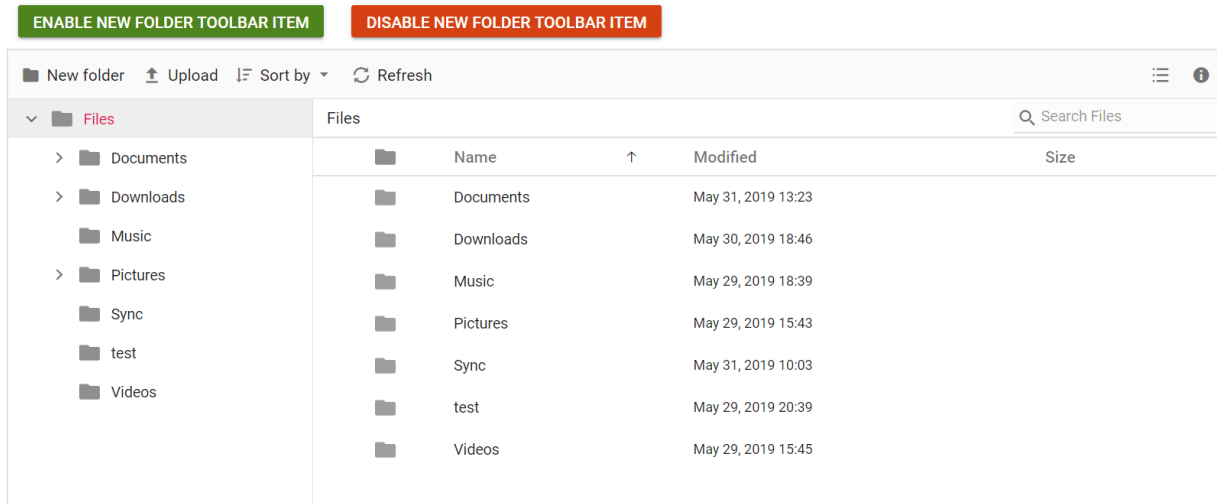
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 }

```

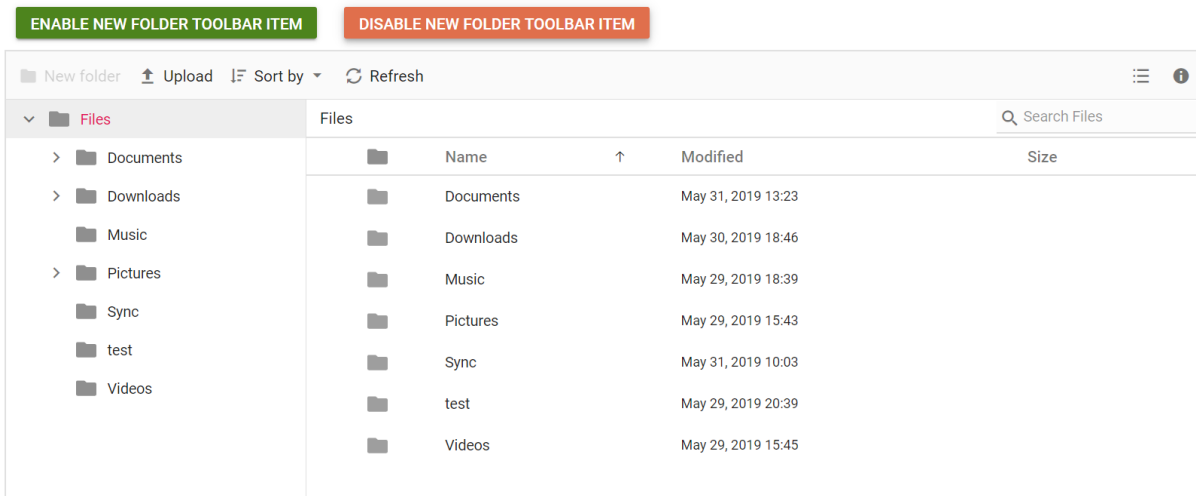
```
 return null;
 }
 // Processing the Upload operation
 public ActionResult Upload(string path,
 IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
 {
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);

 return Content("");
 }
 // Processing the Download operation
 public ActionResult Download(string downloadInput)
 {
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
 }
 // Processing the GetImage operation
 public ActionResult GetImage(FileManagerDirectoryContent args)
 {
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
 }
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below when enable toolbar item.



Output be like the below when enable toolbar item.



### How to add custom thumbnail in file manager

The default appearance of the file manager can customize with your own icon by using [showThumbnail](#) property.

The following example demonstrate how to add a custom icon in largeicons view.

#### CSHTML

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->

 @Html.EJS().FileManager("filemanager").ShowThumbnail(false).AjaxSettings(new
 Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
```



```
UploadUrl = "/Home/Upload",
DownloadUrl = "/Home/Download"
}).Render()
<!-- end of filemanager element -->
</div>
</div>
<style>
.e-filemanager .e-large-icons .e-fe-image {
 background-image:
url("data:image/svg+xml;base64,PD94bWwgdmlvY2lvdjB0Ims4wIIblbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnpB249IjEuMSIGaWQ9IkxheWVYXzEiIHhtbG5zPSJodHRwOi8vd3d3LnclL
m9yZy8yMDAwL3N2ZyIgeG1sbmM6eGxpbnMs9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbnSiI
Hg9IjBweCIgeT0iMHBM4IiB2aWV3Qm94PSiwIDAgMzIgMzIiIHNoeWxlPSJlbmFibGUtYmFja2dyb
3VuZDpuZXcgMCAwIDMyIDMyOyIgeG1sOnNwYWwNLSJwcwVzZXJ2ZSI+PHNoeWxlIHR5cGU9InRle
HQvY3NzIj4uc3Qwe2ZpbGw6IOZGOTI5Mjt9LnN0MXtmaWxsOiNFODdFN0U7fS5zdDJ7ZmlsbDoJR
kZDM0Mz030uc3Qze2ZpbGw6IzkxRDGRGTt9LnN0NHtmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDojQ
zFFN0ZGO30uc3Q2e2ZpbGw6IOZGRkZGRjt9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDojN
DZDNjhDO30uc3Q5e2ZpbGw6IOJCRTHeODt9LnN0MTB7ZmlsbDojRkVCMTdEO30uc3QxMXtmaWxsO
iNERDk2NjY7fS5zdDEye2ZpbGw6IOZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMEkyO30uc3QxNHtma
WxsOiNGMUM1QzU7fS5zdDE1e2ZpbGw6IOZRCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3Qk7fS5zdDE4e2ZpbGw6IOZFRQ0VDRTt9LnN0MT17ZmlsbDojQjdCN0I3O30uc
3QyMHtmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO
30uc3QyM3tmaWxsOiNFNKERTg7fS5zdDIEe2ZpbGw6IOQ2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZF030uc3QyNntmaWxsOiM5OENFNUY7fS5zdDIE3e2ZpbGw6IzhDUYyQzt9LnN0Mjh7ZmlsbDojQ
zFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6IOVEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEEXRUYY7fS5zdDMze2ZpbGw6IOREODDERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNB0EEYRjQ7fS5zdDM2e2ZpbGw6Izg0ODVFODt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHtmaWxsOiNCQ0JDQkm7fS5zdDM5e2ZpbGw6IOE4QThBODt9L
nN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHtmaWxsOiNGRkI3QTQ7fS5zdDQ1e2ZpbGw6IOY2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
zhEQzk3Nzt9LnN0ND17ZmlsbDojN0NBODUXO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGR
kZGO308L3N0eWxlPjxnPjxnPjxwYXRoIGNsYXNzPSJzdDAiIGQ9Ik0yMS41LDAuNWgtMTdjLTEuM
SwLTIsMC45LTIsMnYyNmMwLDEuMSwwLjksMiwyLDJoMjNjMS4xLDAsMi0wLjksMi0ydi0yMUWyM
S41LDAuNXoiLz48cGF0aCBjbGFzc0ic3QxIiBkPSJNMjcunSWzMmgtMjNDMy4xLDMyLDIsMzAuO
SwyLDI5LjV2LTI3QzIsMS4xLDMuMSwwLjDQuNSwwaDE3LjJMMzAsOC4zdjIxLjJDMzAsMzAuOSwyO
C45LDMYLDI3LjUsMzJ6IE00LjUsMUMzLjcsMSwzLDEuNywzLDIuNXxyYn0MZLDMwLjJsMy43LDMXL
DQuNSwzMWgyM2MwLjJgsMCwxLjUtMC43LDEuNS0xLjVWOC43TDIxLjJsMuUgOljV6Ii8+PC9nPjxnP
jxwYXRoIGNsYXNzPSJzdDIiIGQ9Ik0yMS41LDAuNXxyYzAsMC42LDAuNCwxLDESMMwg3TDIxLjUsM
C4leiIvPjxwYXRoIGNsYXNzPSJzdDEiIGQ9Ik0yOS41LDl0LTDdmJEUyNyw5LDIxLDguMywyMSw3L
jV2LTdjMCO30uc3QzISMC4xLTAuNCwwLjMtMC41YzAuMi0wLjEsMC40LDAuSMC41LDAuMWw4LDhDMzAsO
C4zLDMwLjDguNSwzMw4LjldDMjkuOSw4LjksMjkuNyw5LDI5LjUsOXogTTIyLDEuN3Y1LjhhDMjIsN
y44LDIyLjIsOCwyMi41LDhoNS44TDIyLDEuN3oiLz48L2c+PC9nPjxnPjxwb2x5Z29uIGNsYXNzP
SJzdDYiIHBMvaW50cz0iNSwyNCaxMC42LDE3LjJlIGTMQuOCwyMS44IDIxLjQsMTMGmjcsMjQgIi8+P
C9nPjwvc3ZnPg==");
}
.e-filemanager .e-large-icons .e-fe-music {
 background-image:
url("data:image/svg+xml;base64,PD94bWwgdmlvY2lvdjB0Ims4wIIblbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnpB249IjEuMSIGaWQ9IkxheWVYXzEiIHhtbG5zPSJodHRwOi8vd3d3LnclL
m9yZy8yMDAwL3N2ZyIgeG1sbmM6eGxpbnMs9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbnSiI
Hg9IjBweCIgeT0iMHBM4IiB2aWV3Qm94PSiwIDAgMzIgMzIiIHNoeWxlPSJlbmFibGUtYmFja2dyb
3VuZDpuZXcgMCAwIDMyIDMyOyIgeG1sOnNwYWwNLSJwcwVzZXJ2ZSI+PHNoeWxlIHR5cGU9InRle
HQvY3NzIj4uc3Qwe2ZpbGw6IOZGOTI5Mjt9LnN0MXtmaWxsOiNFODdFN0U7fS5zdDJ7ZmlsbDoJR
kZDM0Mz030uc3Qze2ZpbGw6IzkxRDGRGTt9LnN0NHtmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDojQ
zFFN0ZGO30uc3Q2e2ZpbGw6IOZGRkZGRjt9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDojN
DZDNjhDO30uc3Q5e2ZpbGw6IOJCRTHeODt9LnN0MTB7ZmlsbDojRkVCMTdEO30uc3QxMXtmaWxsO
iNERDk2NjY7fS5zdDEye2ZpbGw6IOZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMEkyO30uc3QxNHtma
WxsOiNGMUM1QzU7fS5zdDE1e2ZpbGw6IOZRCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3Qk7fS5zdDE4e2ZpbGw6IOZFRQ0VDRTt9LnN0MT17ZmlsbDojQjdCN0I3O30uc
3QyMHtmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO
30uc3QyM3tmaWxsOiNFNKERTg7fS5zdDIEe2ZpbGw6IOQ2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZF030uc3QyNntmaWxsOiM5OENFNUY7fS5zdDIE3e2ZpbGw6IzhDUYyQzt9LnN0Mjh7ZmlsbDojQ
zFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6IOVEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEEXRUYY7fS5zdDMze2ZpbGw6IOREODDERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNB0EEYRjQ7fS5zdDM2e2ZpbGw6Izg0ODVFODt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHtmaWxsOiNCQ0JDQkm7fS5zdDM5e2ZpbGw6IOE4QThBODt9L
nN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHtmaWxsOiNGRkI3QTQ7fS5zdDQ1e2ZpbGw6IOY2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
zhEQzk3Nzt9LnN0ND17ZmlsbDojN0NBODUXO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGR
kZGO308L3N0eWxlPjxnPjxnPjxwYXRoIGNsYXNzPSJzdDAiIGQ9Ik0yMS41LDAuNWgtMTdjLTEuM
SwLTIsMC45LTIsMnYyNmMwLDEuMSwwLjksMiwyLDJoMjNjMS4xLDAsMi0wLjksMi0ydi0yMUWyM
S41LDAuNXoiLz48cGF0aCBjbGFzc0ic3QxIiBkPSJNMjcunSWzMmgtMjNDMy4xLDMyLDIsMzAuO
SwyLDI5LjV2LTI3QzIsMS4xLDMuMSwwLjDQuNSwwaDE3LjJMMzAsOC4zdjIxLjJDMzAsMzAuOSwyO
C45LDMYLDI3LjUsMzJ6IE00LjUsMUMzLjcsMSwzLDEuNywzLDIuNXxyYn0MZLDMwLjJsMy43LDMXL
DQuNSwzMWgyM2MwLjJgsMCwxLjUtMC43LDEuNS0xLjVWOC43TDIxLjJsMuUgOljV6Ii8+PC9nPjxnP
jxwYXRoIGNsYXNzPSJzdDIiIGQ9Ik0yMS41LDAuNXxyYzAsMC42LDAuNCwxLDESMMwg3TDIxLjUsM
C4leiIvPjxwYXRoIGNsYXNzPSJzdDEiIGQ9Ik0yOS41LDl0LTDdmJEUyNyw5LDIxLDguMywyMSw3L
jV2LTdjMCO30uc3QzISMC4xLTAuNCwwLjMtMC41YzAuMi0wLjEsMC40LDAuSMC41LDAuMWw4LDhDMzAsO
C4zLDMwLjDguNSwzMw4LjldDMjkuOSw4LjksMjkuNyw5LDI5LjUsOXogTTIyLDEuN3Y1LjhhDMjIsN
y44LDIyLjIsOCwyMi41LDhoNS44TDIyLDEuN3oiLz48L2c+PC9nPjxnPjxwb2x5Z29uIGNsYXNzP
SJzdDYiIHBMvaW50cz0iNSwyNCaxMC42LDE3LjJlIGTMQuOCwyMS44IDIxLjQsMTMGmjcsMjQgIi8+P
C9nP
```

```
iNERDk2NjY7fS5zdDEye2ZpbGw6I0ZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMkEyO30uc3QxNHTma
WxsOiNGMUM1QzU7fS5zdDE1e2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRtT9LnN0MTl7ZmlsbDojQjdCN0I3O30uc
3QyMHTmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO
30uc3QyM3tmaWxsOiNFNkE2RTg7fS5zdDI0e2ZpbGw6I0Q2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZF030uc3QyNntmaWxsOiM5OENFNUY7fS5zdDI3e2ZpbGw6IzhDQUYyQzt9LnN0Mjh7ZmlsbDojQ
zZFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6I0VEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEExRUY7fS5zdDMze2ZpbGw6I0REODdERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNB0EEYRjQ7fS5zdDM2e2ZpbGw6Izg4ODVFODt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHTmaWxsOiNCQ0JDQkM7fS5zdDM5e2ZpbGw6I0E4QThBODt9L
nN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHTmaWxsOiNGRkI3QTQ7fS5zdDQ1e2ZpbGw6I0Y2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
zhEQzk3Nzt9LnN0NDl7ZmlsbDojN0NBODUxO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGR
kZGO308L3N0eWxlPjxnPjxwYXR0IGNsYXNzPSJzdDmiIGQ9Ik0yMS41LDAuNWgtMTdjLTEuMSwwL
TIsMCA45LTIsMnYyN2MwLDEuMSwwLjksMiwYLDJoMjNjMS4xLDAsMi0wLjksMi0ydi0yMUwyMS41L
DAuNXoiLz48cGF0aCBjbGFzc20ic3QyMSIgZD0iTTI3LjUsMzJoLTIZQzMuMSwzMiwyLDMwLjksM
iwyOS41di0yN0MyLDEuMSwzLjEsMCAwLjUsMGgxNy4yTDMwLDguM3YyMS4yQzMwLDMwLjksMjguO
SwzMiwyNy41LDMyeiBNNC41LDFDMY43LDEsMywxLjcsMywyLjV2MjdDMywwZC4zLDMuNywwZC4zL
jUsMzFoMjNjMCA44LDAsMS41LTAuNywxLjUtMS41VjguN0wyMS4zLDFINC41eiIvPjwvZz48Zz48c
GF0aCBjbGFzc20ic3Q1IiBkPSJNMjEuNSwwLjV2N2MwLDAuNiwwLjQsMSwzLDFoN0wyMS41LDAuN
XoiLz48cGF0aCBjbGFzc20ic3QyMSIgZD0iTTI5LjUsOWgtN0MyMS43LDksMjEsOC4zLDIxLDCuN
XYtN2MwLTAuMiwwLjEtMCA40LDAuMy0wLjVjMCA4yLTAuMSwwLjQsMCAwLjUsMCA4xbDgsOEMzMCw4L
jMsMzAsOC41LDMwLDguN0MyOS45LDguOSwyOS43LDksMjkuNSw5eiBNMjIsMS43djUuOEMyMiwwL
jgsMjIuMiwwLjUsOGg1LjhmMjIsMS43eiIvPjwvZz48Zz48cGF0aCBjbGFzc20ic3Q2IiBkP
SJNMTIsMTQuOXy3LjZjLTAuNi0wLjUtMS40LTAuNy0yLjMtMCA40Yy0wLjgsMCA4yLDEuNCwwLjktM
S42LDEuN2MtMCA41LDEuOCwxLDMuNSwyLjgsMy4yYzEuMi0wLjIsMi4xLDEuNCwyLjEtMi42bDatN
i40bDEwLDEuOXY0LjNjLTAuNi0wLjUtMS40LTAuNy0yLjMtMCA40Yy0wLjgsMCA4yLDEuNCwwLjktM
S42LDEuN2MtMCA41LDEuOCwxLDMuNSwyLjgsMy4yYzEuMi0wLjIsMi4xLDEuNCwyLjEtMi42bDatO
S4zYzAtMCA42LTAuNi0xLjEtMS4yLTFsLDEwLDEuOUMxMi4zLDE0LDEyLDE0LjQsMTIsMTQuOXoiL
z48L2c+PC9zdmc+");
```

```
}
.e-filemanager .e-large-icons .e-fe-xlsx {
 background-image:
```

```
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnNpb249IjEuMSIgawQ9IkxheWVYXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczL
m9yZy8yMDAwL3N2ZyIgeG1sbnM6Gxpbms9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbmsiI
Hg9IjBweCIgeT0iMHB4IiB2aWV3Qm94PSIwIDAuMzIuMzIuMzIuMzIuMzIuMzIuMzIuMzIuMzIuM
3VuZDpuZXCgMCAwIDMyOyIgeG1sOnNwYWNlPSJwcmVzZXJ2ZSI+PHN0eWxlIHR5cGU9InRle
HQvY3NzIj4uc3Qwe2ZpbGw6I0ZGOTI5Mj9LnN0MTN7ZmlsbDojRjJBMkEyO30uc3QxNHTma
WxsOiNGMUM1QzU7fS5zdDE1e2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRtT9LnN0MTl7ZmlsbDojQjdCN0I3O30uc
3QyMHTmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO
30uc3QyM3tmaWxsOiNFNkE2RTg7fS5zdDI0e2ZpbGw6I0Q2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZF030uc3QyNntmaWxsOiM5OENFNUY7fS5zdDI3e2ZpbGw6IzhDQUYyQzt9LnN0Mjh7ZmlsbDojQ
zZFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6I0VEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEExRUY7fS5zdDMze2ZpbGw6I0REODdERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNB0EEYRjQ7fS5zdDM2e2ZpbGw6Izg4ODVFODt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHTmaWxsOiNCQ0JDQkM7fS5zdDM5e2ZpbGw6I0E4QThBODt9L
nN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHTmaWxsOiNGRkI3QTQ7fS5zdDQ1e2ZpbGw6I0Y2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
zhEQzk3Nzt9LnN0NDl7ZmlsbDojN0NBODUxO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGR
```

2685

Copyright © 2001 -2024 Syncfusion Inc.

```
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnNpb249IjEuMSIgaWQ9IkxheWVyXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczLm
y9yZy8yMDAwL3N2ZyIgeG1sbmM6eGxpbnMsImh0dHA6Ly93d3duc2Mub3JnLnZlE5OTkveGxpbnMiI
Hg9IjBweCIgeT0iMHBM4iIB2aWV3Qm94PSIwIDAgMzIgMzIiIHNoeWxlPSJlbmFibGUtYmFja2dyb
3VuZDpuZXcgMCAwIDMyIDMyOyIgeG1sOnNwYWYWNlPSJwcmlvZmVzcXJ2ZSI+PHNoeWxlIHR5cGU9InRl
eHQvY3NzIj4uc3Qwe2ZpbGw6IOZGOTI5Mjt9LnN0MXtmaWxsOiNFODdFN0U7fS5zdDJ7ZmlsbDoJR
kZDM0MzO3uc3Qze2ZpbGw6IzkxRDGRGTt9LnN0NHtmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDoJJ
zFFNOZGO30uc3Q2e2ZpbGw6IOZGRkZGRjdt9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDoJN
dZDNjdDO30uc3Q5e2ZpbGw6IOJCRtZGErdt9LnN0MTB7ZmlsbDoJRkVCMTEfS0uc3Q3MXtmaWxsOi
NERdk2NjY7fS5zdDEye2ZpbGw6IOZFDRDCNzt9LnN0MTN7ZmlsbDoJRjBJMKEyO30uc3Q3NXHtma
WxsOiNGMUUMIQzU7fS5zdDEle2ZpbGw6IORCQiy2Mzt9LnN0MTZ7ZmlsbDoJQ0VBMTUXo30uc3Q3XN
```

```
.e-filemanager .e-large-icons .e-fe-txt {
 background-image:
```

---

Copyright © 2001 -2024 Syncfusion Inc.

2689

```
y0wLjMtMS0wLjRzLTAuNS0wLjMtMC43LTAuNWMtMC4yLTAuMi0wLjMtMC40LTAuNC0wLjdTMTYsM
TcuNSwxNiwxNy4yYzAtMC44LDAuMi0xLjUsMC43LTEuOXMxLjEtMC42LDEuOS0wLjZjMC40LDAuM
C43LDAuMS4xLDAuMWMwLjMsMC4xLDAuNiwwLjIsMC44LDAuNGMwLjIsMC4yLDAuNCwwLjQsMC42L
DAuN2MwLjEsMC4zLDAuMiwwLjYsMC4yLDF2MC4yaC0xLjdjMC0wLjQtMC4xLTAuNy0wLjItMC45Y
y0wLjEtMC4yLTAuNC0wLjMtMC43LTAuM2MtMC4yLDAuTMC40LDAuTMC41LDAuMVMxOCwxNi4xLDE4L
DE2LjJjLTAuMSwwLjEtMC4xLDAuMi0wLjIsMC4zYzAsMC4xLDAuMS4yLDAuMS40YzAsMC4zLDAuM
SwwLjUsMC4yLDAuN2MwLjEsMC4yLDAuNCwwLjMsMC43LDAuNWMwLjMsMC42YzAuMywwLjEsMC42L
DAuMywwLjgsMC40czAuNCwwLjMsMC41LDAuNXMwLjIsMC40LDAuMywwLjZjMCwwLjIsMC4xLDAuN
SwwLjEsMC43YzAsMC45LTAuMywwLjYtMC44LDJjLTAuNSwwLjQtMS4zLDAuNi0yLjIsMC42Yy0xL
DAuTMS43LTAuMi0yLjEtMC42Yy0wLjQtMC40LTAuNi0xLTAuNi0xLjh2LTAuM2gxLjdWMjF6Ii8+P
C9nPjwvZz48L3N2Zz4=");
}
.e-filemanager .e-large-icons .e-fe-css {
 background-image:
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnNpb249IjEuMSIgaWQ9IkkxheWVYXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczL
m9yZy8yMDAwL3N2ZyIgeG1sbnM6eGxpbnM9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbnMsiI
Hg9IjBweCIgeT0iMHB4IiB2aWV3Qm94PSIwIDAgaWQ9IjEiIHNoeWx1PSJlbmFibGUtYmFja2dyb
3VuZDpuZXcgMCAwIDMyOyIgeG1sOnNwYWN1PSJwcmVzZXJ2ZSI+PHN0eWx1IHR5cGU9InRle
HQvY3NzIj4uc3Qwe2ZpbGw6I0ZGOTI5Mjt9LnN0MXTmaWxsOiNFODdFN0U7fS5zdDJ7ZmlsbDojR
kZDM0MzO30uc3Qze2ZpbGw6IzkrRDRGRt9LnN0NHTmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDojQ
zFFN0ZGO30uc3Q2e2ZpbGw6I0ZGRkZGRj9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDojN
DZDNjhDO30uc3Q5e2ZpbGw6I0JCRThEODt9LnN0MTB7ZmlsbDojRkVCMTdEO30uc3QxMXtmaWxsO
iNERDk2NjY7fS5zdDEye2ZpbGw6I0ZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMkEyO30uc3QxNHTma
WxsOiNGMUM1QzU7fS5zdDEle2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRt9LnN0MT17ZmlsbDojQjdCN0I3O30uc
3QyMHTmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO
30uc3QyM3tmaWxsOiNFNke2RTg7fS5zdDI0e2ZpbGw6I0Q2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZFO30uc3QyNntmaWxsOiM5OENFNUY7fS5zdDI3e2ZpbGw6IzhDUYyQzt9LnN0Mjh7ZmlsbDojQ
zZFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6I0VEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEExRUY7fS5zdDMze2ZpbGw6I0REODERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNB0EEYRjQ7fS5zdDM2e2ZpbGw6Izg4ODVfODt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHTmaWxsOiNCQ0JDQkM7fS5zdDM5e2ZpbGw6I0E4QThBODt9L
nN0ND7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRfM0VCO30uc3Q0NHTmaWxsOiNGRkI3QTQ7fS5zdDQle2ZpbGw6I0Y2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
zhEQzk3Nzt9LnN0ND17ZmlsbDojN0NBODUxO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGR
kZGO308L3N0eWx1PjxnPjxwYXR0IGNsYXNzPSJzdDI5IiBkPSJNMjEuNSwwLjVoLTE3Yy0xLjEsM
C0yLDAuOS0yLDJ2MjdjMCwwLjEsMC45LDI5MiwyADIzYzEuMSwwLDItMC45LDItMnYtMjFMMjEuN
SwwLjV6Ii8+PHBhdGggY2xhc3M9InN0MzAiIGQ9Ikk0Yy41LDMyaC0yM0MzLjEsMzIsMiwwMC45L
DI5MjkuNXItMjdmIiwLjEsMy4xLDAuNS41LDBoMTcuMkwzMCw4LjN2MjEuMkMzMCwzMC45LDI4L
jksMzIsMjcuNSwzMnogTTQuNSwzQzMuNywxLDMsMS43LDMsMi41djI3QzMsMzAuMywzLjcsMzEsN
C41LDMxaDIzYzAuOCwwLDEuNS0wLjcsMS41LTEuNVY4LjdMMjEuMywxSDQuNXoiLz48L2c+PGc+P
HBhdGggY2xhc3M9InN0MzEiIGQ9Ikk0YyMS41LDAuNXy3YzAsMC42LDAuNCwwLDEsMWg3TDIxLjUsM
C41eiIvPjxwYXR0IGNsYXNzPSJzdDMwIiBkPSJNMjkuNSw5aC03QzIxLjcsOSwyMSw4LjMsMjEsN
y41di03YzAtMC4yLDAuMS0wLjQsMC4zLTAuNWMwLjItMC4xLDAuNCwwLDAuNSwwLjFsOCw4QzMwL
DguMywzMCw4LjUsMzAsOC43QzI5LjksOC45LDI5LjcsOSwyOS41LDl6IE0yMiwwLjd2NS44QzIyL
DcuOCwyMi4yLDgsMjIuNSw4aDUuOEwyMiwwLjd6Ii8+PC9nPjxnPjxwYXR0IGNsYXNzPSJzdDYiI
GQ9Ikk0xMC43LDE2LjNjLTAuMS0wLjMtMC40LTAuNC0wLjgtMC40Yy0wLjIsMC0wLjQsMC4xLTAuN
SwwLjJjLTAuMSwwLjEtMC4yLDAuMy0wLjMsMC42cy0wLjEsMC42LTAuMiwwYzAsMC40LDAuMS45L
DAsMS41YzAsMC42LDAuMS4yLDAuMSwwLjZzMC4xLDAuNywwLjIsMC45czAuMiwwLjQsMC4zLDAuN
GMwLjEsMC4xLDAuMywwLjEsMC41LDAuMWMwLjIsMCwwLjMsMCwwLjQtMC4xYzAuMS0wLjEsMC4yL
TAuMiwwLjMtMC4zYzAuMS0wLjIsMC4yLTAuNCwwLjItMC43YzAuMS0wLjMsMC4xLTAuNywwLjEtM
S4xaDEuN2MwLDAuNSwwLDAuOS0wLjEsMS4zcy0wLjIsMC44LTAuNCwwLjFjLTAuMiwwLjMtMC41L
DAuNi0wLjgsMC43Yy0wLjQsMC4yLTAuOCwwLjMtMS40LDAuM2MtMC42LDAuTMS4yLTAuMS0xLjUtM
C4zYy0wLjQtMC4yLTAuNy0wLjUtMC45LTAuOXMtMC4zLTAuOS0wLjQtMS40Yy0wLjEtMC41LTAuM
S0xLjEtMC4xLTEuOGMwLTAuNiwwLTAuMiwwLjEtMS44YzAuMS0wLjUsMC4yLTEsMC40LTEuNGMwL
```



```
jItMC40LDAuNS0wLjcsMC45LTAuOWMwLjQtMC4yLDAuOS0wLjMsMS41LTAuM2MwLjYsMCwxLjEsM
C4xLDEuNSwwLjNjMC40LDAuMiwwLjYsMC41LDAuOCwwLjhzMC4zLDAuNiwwLjMsMwMwLDAuNCwwL
jEsMC43LDAuMSwxaC0xLjdMTAuOSwxNy4xLDEwLjgsMTYUuNiwxMC43LDE2LjN6Ii8+PHBhdGggY
2xhc3M9InN0NiIgZD0iTTE1LjEsMjFjMCwwLjIsMCwwLjQsMCwwLjVzMC4xLDAuMywwLjIsMC40Y
zAuMSwwLjEsMC4yLDAuMiwwLjQsMC4zYzAuMSwwLjEsMC4zLDAuMSwwLjYsMC4xYzAuMywwLDAuN
S0wLjEsMC43LTAuM3MwLjMtMC40LDAuMy0wLjhjMC0wLjIsMC0wLjQtMC4xLTAuNXMtMC4xLTAuM
y0wLjMtMC40Yy0wLjEtMC4xLTAuMy0wLjItMC41LTAuM2MtMC4yLTAuMS0wLjQtMC4yLTAuNy0wL
jNjLTAuNC0wLjEtMC43LTAuMy0xLTAuNFMxNC4yLDE5LDE0LDE4LjhjLTAuMi0wLjItMC4zLTAuN
C0wLjQtMC43cy0wLjEtMC42LTAuMS0wLjljMC0wLjgsMC4yLTEuNSwwLjctMS45czEuMS0wLjYsM
S45LTAuNmMwLjQsMCwwLjcsMCwxLjEsMC4xYzAuMywwLjEsMC42LDAuMiwwLjgsMC40YzAuMiwwL
jIsMC40LDAuNCwwLjYsMC43YzAuMSwwLjMsMC4yLDAuNiwwLjIsMXyYwLjJoLTEuN2MwLTAuNC0wL
jEtMC43LTAuMi0wLjljLTAuMS0wLjItMC40LTAuMy0wLjctMC4zYy0wLjIsMC0wLjQsMC0wLjUsM
C4xcy0wLjIsMC4xLTAuMywwLjJjLTAuMSwwLjEtMC4xLDAuMi0wLjIsMC4zYzAsMC4xLDAsMC4yL
DAsMC40YzAsMC4zLDAuMSwwLjUsMC4yLDAuN2MwLjEsMC4yLDAuNCwwLjMsMC43LDAuNWwxLjMsM
C42YzAuMywwLjEsMC42LDAuMywwLjgsMC40czAuNCwwLjMsMC41LDAuNVMxOSwyMCwxOSwyMC4yY
zAsMC4yLDAuMSwwLjUsMC4xLDAuN2MwLDAuOS0wLjMsMS42LTAuOCwyYy0wLjUsMC40LTEuMywwL
jYtMi4yLDAuNmMtMSwwLTEuNy0wLjItMi4xLTAuNmMtMC40LTAuNC0wLjYtMS0wLjYtMS44di0wL
jNoMS43VjIxeiIvPjxwYXRoIGNsYXNzPSJzdDYiIGQ9Ik0yMS4zLDIxYzAsMC4yLDAsMC40LDAsM
C41czAuMSwwLjMsMC4yLDAuNGMwLjEsMC4xLDAuMiwwLjIsMC40LDAuM2MwLjEsMC4xLDAuMywwL
jEsMC42LDAuMwMwLjMsMCwwLjUtMC4xLDAuNy0wLjNzMC4zLTAuNCwwLjMtMC44YzAtMC4yLDAuT
C40LTAuMS0wLjVzLTAuMS0wLjMtMC4zLTAuNGMtMC4xLTAuMS0wLjMtMC4yLTAuNS0wLjNjLTAuM
i0wLjEtMC40LTAuMi0wLjctMC4zYy0wLjQtMC4xLTAuNy0wLjMtMS0wLjRzLTAuNS0wLjMtMC43L
TAuNwMtMC4yLTAuMi0wLjMtMC40LTAuNC0wLjdzLTAuMS0wLjYtMC4xLTAuOWMwLTAuOCwwLjItM
S41LDAuNy0xLjlzMS4xLTAuNiwxLjktMC42YzAuNCwwLDAuNywwLDEuMSwwLjFjMC4zLDAuMSwwL
jYsMC4yLDAuOCwwLjRjMC4yLDAuMiwwLjQsMC40LDAuNiwwLjdjMC4xLDAuMywwLjIsMC42LDAuM
iwxLjAuMmgtMS43YzAtMC40LTAuMS0wLjctMC4yLTAuOWMtMC4xLTAuMi0wLjQtMC4zLTAuNy0wL
jNjLTAuMiwwLTAuNCwwLTAuNSwwLjFzLTAuMiwwLjEtMC4zLDAuMmMtMC4xLDAuMS0wLjEsMC4yL
TAuMiwwLjNjMCwwLjEsMCwwLjIsMCwwLjRjMCwwLjMsMC4xLDAuNSwwLjIsMC43YzAuMSwwLjIsM
C40LDAuMywwLjcsMC41bDEuMywwLjZjMC4zLDAuMSwwLjYsMC4zLDAuOCwwLjRzMC40LDAuMywwL
jUsMC41czAuMiwwLjQsMC4zLDAuNmMwLDAuMiwwLjEsMC41LDAuMSwwLjdjMCwwLjktMC4zLDEuN
i0wLjgsMmMtMC41LDAuNC0xLjMsMC42LTIuMiwwLjZjLTEsMC0xLjctMC4yLTIuMS0wLjZjLTAuN
C0wLjQtMC42LTEtMC42LTEuOHYtMC4zaDEuN1YyMXoiLz48L2c+PC9zdmc+");
}
.e-filemanager .e-large-icons .e-fe-html {
 background-image:
url("data:image/svg+xml;base64,PD94bWwgdmVyc2l1bWw0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnNpb249IjEuMSIgaWQ9IkxheWVYXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczL
m9yZy8yMDAwL3N2ZyIgeG1sbnM6eGxpbnM9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbnMsiI
Hg9IjBweCIgeT0iMHB4IiB2aWV3Qm94PSIwIDAgaWQ9IiIHN0eWx1PSJlbmFibGUtYmFja2dyb
3VuZDpuZxcgMCAwIDMyOyIgeG1sOnNwYWN1PSJwcmVzZXJ2ZSI+PHN0eWx1IHR5cGU9InRle
HQvY3NzIj4uc3Qwe2ZpbGw6I0ZGOTI5Mj9LnN0MXtmaWxsOiNFODdFN0U7fS5zdDJ7ZmlsbDojR
kZDM0Mz030uc3Qze2ZpbGw6IzIxRDRGRt9LnN0NHTmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDojQ
zFFN0ZG030uc3Q2e2ZpbGw6I0ZGRkZGRj9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDojN
DZDNjhDO30uc3Q5e2ZpbGw6I0JCRThEODt9LnN0MTB7ZmlsbDojRkVCMTdEO30uc3QxMXtmaWxsO
iNERDk2NjY7fS5zdDEye2ZpbGw6I0ZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMkEyO30uc3QxNHTma
WxsOiNGMUM1QzU7fS5zdDEle2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRTt9LnN0MTl7ZmlsbDojQjdcN0I3O30uc
3QyMHTmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTDBo
30uc3QyM3tmaWxsOiNFNkE2RTg7fS5zdDI0e2ZpbGw6I0Q2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZFO30uc3QyNntmaWxsOiM5OENFNUY7fS5zdDI3e2ZpbGw6IzhDQUYyQzt9LnN0Mjh7ZmlsbDojQ
zZFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6I0VEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEExRUY7fS5zdDMze2ZpbGw6I0REODdERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNB0EEYRjQ7fS5zdDM2e2ZpbGw6Izg4ODVFODt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHTmaWxsOiNCQ0JDQkM7fS5zdDM5e2ZpbGw6I0E4QThBODt9L
nN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHTmaWxsOiNGRkI3QTQ7fS5zdDQle2ZpbGw6I0Y2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
```

2692

---

Copyright © 2001 -2024 Syncfusion Inc. 2693

```
WxsOiNGMUM1QzU7fS5zdDE1e2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRtT9LnN0MTl7ZmlsbDojQjdCN0I3O30uc
3QyMHTmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO
30uc3QyM3tmaWxsOiNfNkE2RTg7fS5zdDI0e2ZpbGw6I0Q2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ
0ZF030uc3QyNntmaWxsOiM5OENFNUY7fS5zdDI3e2ZpbGw6IzhDQUYyQzt9LnN0Mjh7ZmlsbDojQ
zZFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6I0VEOUY2NDt9LnN0MzF7Zmlsb
DojRkZENkI1O30uc3QzMntmaWxsOiNGNEExRUY7fS5zdDMze2ZpbGw6I0REODdERDt9LnN0MzR7Z
mlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNBOEEYRjQ7fS5zdDM2e2ZpbGw6Izg4ODVFOdt9LnN0M
zd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHTmaWxsOiNCQ0JDQkM7fS5zdDM5e2ZpbGw6I0E4QThBODt9L
nN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDM
Tt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHTmaWxsOiNGRkI3QTQ7fS5zdDQle2ZpbGw6I0Y2O
UE3Qjt9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6I
zheQzk3Nzt9LnN0NDl7ZmlsbDojN0NBODUxO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGR
kZGO308L3N0eWxlPjxnPjxwYXRoIGNsYXNzPSJzdDQxIiBkPSJNMjEuNSwwLjVoLTE3Yy0xLjEsM
C0yLDAuOS0yLDJ2MjdjMCwLjEsMCA45LDIsMiwyaDIzYzEuMSwwLDItMCA45LDItMnYtMjFMMjEuN
SwwLjV6Ii8+PHBhdGggY2xhc3M9InN0NDIiIGQ9Ik0yNy41LDMyaC0yM0MzLjEsMzIsMiwzMC45L
DIsMjkuNXytMjdDMiwxLjEsMy4xLDAsNC41LDBoMTcuMkwzMCw4LjN2MjEuMkMzMCA45LDI4L
jksMzIsMjcuNSwzMnogTTQuNSwxQzMuNywxLDMsMS43LDMsMi41djI3QzMzMzAuMywzLjcsMzEsN
C41LDMxaDIzYzAuOCwwLDEuNS0wLjcsMS41LTEuNVY4LjdMMjEuMywxSDQuNXoiLz48L2c+PGc+P
HBhdGggY2xhc3M9InN0NDMiIGQ9Ik0yMS41LDAuNXy3YzAsMC42LDAuNCwwLDEsMWg3TDIxLjUsM
C41eiIvPjxwYXRoIGNsYXNzPSJzdDQyIiBkPSJNMjkuNSw5aC03QzIxLjcsOSwyMSw4LjMsMjEsN
y41di03YzAtMC4yLDAuMS0wLjQsMC4zLTAuNWMwLjItMC4xLDAuNCwwLDAuNSwwLjFsOCw4QzMwL
DguMywzMCw4LjUsMzAsOC43QzI5LjksOC45LDI5LjcsOSwyOS41LDl6IE0yMiwxLjd2NS44QzIyL
DcuOCwyMi4yLDgsMjIuNSw4aDUuOEwyMiwxLjd6Ii8+PC9nPjxnPjxwYXRoIGNsYXNzPSJzdDYiI
GQ9Ik05LjYsMTQuOGwxLjIsNmgbwDEuMi02aDIuNnY4LjZoLTEuNnYtNi44aDBsLTEuNSw2LjhoL
TEuM2wtMS41LTYuOGgdjYuOEg3LjF2LTguNkg5LjZ6Ii8+PHBhdGggY2xhc3M9InN0NiIgZD0iT
E3LjMsMjFjMCAwLjIsMCAwLjQsMCAwLjVzMC4xLDAuMywwLjIsMCA40YzAuMSwwLjEsMCA4yLDAuM
iwwLjQsMCA4YzAuMSwwLjEsMCA4zLDAuMSwwLjYsMCA4YzAuMywwLDAuNS0wLjEsMCA43LTAuM3MwL
jMtMC40LDAuMy0wLjhjMCAwLjIsMCAwLjQtMC4xLTAuNXMtMC4xLTAuMy0wLjMtMC40Yy0wLjEtM
C4xLTAuMy0wLjItMC41LTAuM2MtMC4yLTAuMS0wLjQtMC4yLTAuNy0wLjNjLTAuNC0wLjEtMC43L
TAuMy0xLTAuNHMtMC41LTAuMy0wLjQtMC41Yy0wLjItMC4yLTAuMy0wLjQtMC40LTAuM3MtMC4xL
TAuNi0wLjEtMC45YzAtMC44LDAuMi0xLjUsMC43LTEuOXMxLjEtMC42LDEuOS0wLjZjMC40LDAuM
C43LDAuMS4xLDAuMWMwLjMsMCA4xLDAuNiwwLjIsMCA44LDAuNGMwLjIsMCA4yLDAuNCwwLjQsMCA42L
DAuN2MwLjEsMCA4zLDAuMiwwLjYsMCA4yLDF2MC4yaC0xLjdjMCAwLjQtMC4xLTAuNy0wLjItMC45Y
y0wLjEtMC4yLTAuNC0wLjMtMC43LTAuM2MtMC4yLDAuMCA40LDAuMCA41LDAuMXMtMC4yLDAuMS0wL
jMsMCA4yYy0wLjEsMCA4xLTAuMSwwLjItMC4yLDAuM2MwLDAuMSwwLDAuMiwwLDAuNGMwLDAuMywwL
jEsMCA41LDAuMiwwLjdjMCA4xLDAuMiwwLjQsMCA4zLDAuNywwLjVsMS4zLDAuNmMwLjMsMCA4xLDAuN
iwwLjMsMCA44LDAuNHMwLjQsMCA4zLDAuNSwwLjVzMC4yLDAuNCwwLjMsMCA42YzAsMC4yLDAuMSwwL
jUsMCA4xLDAuN2MwLDAuOS0wLjMsMS42LTAuOCwyYy0wLjUsMCA40LTEuMywwLjYtMi4yLDAuNmMtM
SwwLTEuNy0wLjItMi4xLTAuNmMtMC40LTAuNC0wLjYtMS0wLjYtMS44di0wLjNoMS43VjIxeiIvP
jxwYXRoIGNsYXNzPSJzdDYiIGQ9Ik0yMi4yLDE0LjhIMjR2OC42aC0xLjdWMTQuOHoiLz48L2c+P
C9zdmc+");
}
.e-filemanager.e-large-icons.e-fe-php {
 background-image:
url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz48c3ZnIHZlcnNpb249IjEuMSIgaWQ9IkxheWVyXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczL
m9yZy8yMDAwL3N2ZyIgeG1sbnM6Gxpbms9Imh0dHA6Ly93d3d3LnczLm9yZy8yMDAwL3N2ZyI
Hg9IjBweCigeT0iMHB4IiB2aWV3Qm94PSIwIDAgaWQ9IjEiIHNoeWxlPSJlbmFibGUTYmFja2dyb
3VuZDpuZXCgMCAwIDMyOyIgeG1sbnM6YXNzPSJwcmVzZXJ2ZSI+PHN0eWxlIHR5cGU9InRle
HQvY3NzIj4uc3Qwe2ZpbGw6I0ZGOTI5Mj9LnN0MTZ7ZmlsbDojRkZDM0MzO30uc3Qze2ZpbGw6Iz
kxRDRGRtT9LnN0NHTmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDojQzFFN0ZGO30uc3Q2e2ZpbGw6I0ZGRkZGRj
t9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDojNDZDNjhDO30uc3Q5e2ZpbGw6I0JCRThEODt9LnN0MTB7ZmlsbDojRkVCMTdEO30uc3QxMXtmaWxsO
iNERDk2NjY7fS5zdDEye2ZpbGw6I0ZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMkEyO30uc3QxNHTma
WxsOiNGMUM1QzU7fS5zdDE1e2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN
3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRtT9LnN0MTl7ZmlsbDojQjdCN0I3O30uc
```

```
.e-filemanager .e-large-icons .e-fe-doc,
.e-filemanager .e-large-icons .e-fe-docx {
 background-image:
```

2696

```
DcuOCwyMi4yLDgsMjIuNSw4aDUuOEwyMiwXljd6Ii8+PC9nPjxnPjxwYXRoIGNsYXNzPSJzdDYiIGQ9Ik03LjcsMTQuOGwXlJEsMi44bDEuMS0yLjhoMS45bC0yLDQuMmwyLjEsNC4zSDEwbC0xLjItMi45bC0xLjIsMi45SDUuNkw3LjgsMTlsLTIiNC4ySDcuN3oiLz48cGF0aCBjbGFzc20ic3Q2IiBkPSJNMTUuMiwXNC44bDEuMiw2aDBsMS4yLTZoMi42djguNmgtMS42di02LjhoMGwtMS41LDYuOGgtMS4zbC0xLjUtNi44aDB2Ni44aC0xLjZ2LTguNkgxNS4yeiIvPjxwYXRoIGNsYXNzPSJzdDYiIGQ9Ik0yMS41LDE0LjhoMS43VjIyaDMuMXyXlJRoLTQuOFYxNC44eiIvPjwvZz48L3N2Zz4=");
}
.e-filemanager .e-large-icons .e-fe-folder {
 background-image:
url('data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48c3ZnIHZlcnNpb249IjEuMSIgaWQ9IkkxheWVYXZeiIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIgeG1sbnM6eGxpbnMs9Imh0dHA6Ly93d3d3cudzMub3JnLzE5OTkveGxpbnMsIiHg9IjBweCIgeT0iMHB4IiB2aWV3Qm94PSIwIDAgaWQ9IiIHN0eWx1PSJlbmFibGUtYmFja2dyb3VuZDpuZXRxcGMAwIDMyIDMyOyIgeG1sOnNwYWNlPSJwcmVzZXJ2ZSI+PHN0eWx1IHR5cGU9InRleHQvY3NzIj4uc3Qwe2ZpbGw6I0ZGOTI5Mjt9LnN0MXtmaWxsOiNFODdFN0U7fS5zdDj7ZmlsbDojRkZDM0Mz030uc3Qze2ZpbGw6IzIxRDRGRt9LnN0NHTmaWxsOiM2M0E3RDM7fS5zdDV7ZmlsbDojQzFFN0ZGO30uc3Q2e2ZpbGw6I0ZGRkZGRjt9LnN0N3tmaWxsOiM4M0Q2Qjk7fS5zdDh7ZmlsbDojNDZDNjhDO30uc3Q5e2ZpbGw6I0JCRThEODt9LnN0MTB7ZmlsbDojRkVCMTdEO30uc3QxMXtmaWxsOiNERDk2NjY7fS5zdDEye2ZpbGw6I0ZFRDRCNzt9LnN0MTN7ZmlsbDojRjJBMkEyO30uc3QxNHTmaWxsOiNGMUM1QzU7fS5zdDEle2ZpbGw6I0RCQjY2Mzt9LnN0MTZ7ZmlsbDojQ0VBMTUxO30uc3QxN3tmaWxsOiNFQkQ3QTk7fS5zdDE4e2ZpbGw6I0NFQ0VDRt9LnN0MTl7ZmlsbDojQjdCN0I3O30uc3QyMHTmaWxsOiNFNEU0RTQ7fS5zdDIxe2ZpbGw6IzY1QUFEMTt9LnN0MjJ7ZmlsbDojRTU3QTdBO30uc3QyM3tmaWxsOiNFNkE2RTg7fS5zdDI0e2ZpbGw6I0Q2OEFENjt9LnN0MjV7ZmlsbDojRkZDQ0ZFO30uc3QyNntmaWxsOiM5OENFNUY7fS5zdDI3e2ZpbGw6IzhDQUYyQzt9LnN0Mjh7ZmlsbDojQzZFM0E3O30uc3QyOXtmaWxsOiNGRkI1Nzg7fS5zdDMwe2ZpbGw6I0VEOUY2NDt9LnN0MzF7ZmlsbDojRkZENkI1O30uc3QzMntmaWxsOiNGNEExRUY7fS5zdDMze2ZpbGw6I0REODdERDt9LnN0MzR7ZmlsbDojRjldQkY2O30uc3QzNXtmaWxsOiNBOEEYRjQ7fS5zdDM2e2ZpbGw6Izg4ODVFOdt9LnN0Mzd7ZmlsbDojQ0ZDQ0Y4O30uc3QzOHTmaWxsOiNCQ0JDQkM7fS5zdDM5e2ZpbGw6I0E4QThBODt9LnN0NDB7ZmlsbDojREFEQURBO30uc3Q0MXtmaWxsOiM3N0NDREI7fS5zdDQye2ZpbGw6IzREQkNDMTt9LnN0NDN7ZmlsbDojQjRFM0VCO30uc3Q0NHTmaWxsOiNGRkI3QTQ7fS5zdDQle2ZpbGw6I0Y2OUE3Qj9LnN0NDZ7ZmlsbDojRkZEN0NEO30uc3Q0N3tmaWxsOiM3MUM4RjQ7fS5zdDQ4e2ZpbGw6IzhEQzk3Nzt9LnN0NDl7ZmlsbDojN0NBODUxO30uc3Q1MHTvcGFjaXR5OjAuNDU7ZmlsbDojRkZGRkZGO308L3N0eWx1PjxnPjxwYXRoIGNsYXNzPSJzdDmiIGQ9Ik0yOS41LDI3LjVoLTI3Yy0xLjEsMC0yLTAuOS0yLTJ2LTE5YzAtMS4xLDAuOS0yLDItMmgxMCA40bDMuNSwzLjFoMTMuMmMxLjEsMCAwYLDJ2MTUuOUMzMS41LDI2LjYsMzAuNiwyNy41LDI5LjUsMjcuNXoiLz48cGF0aCBjbGFzc20ic3Q0IiBkPSJNMTUuMiwXNC44bDEuMS0yLjhoMS45bC0yLDQuMmwyLjEsNC4zSDEwbC0xLjItMi45bC0xLjIsMi45SDUuNkw3LjgsMTlsLTIiNC4ySDcuN3oiLz48cGF0aCBjbGFzc20ic3Q2IiBkPSJNMTUuMiwXNC44bDEuMiw2aDBsMS4yLTZoMi42djguNmgtMS42di02LjhoMGwtMS41LDYuOGgtMS4zbC0xLjUtNi44aDB2Ni44aC0xLjZ2LTguNkgxNS4yeiIvPjxwYXRoIGNsYXNzPSJzdDYiIGQ9Ik0yMS41LDE0LjhoMS43VjIyaDMuMXyXlJRoLTQuOFYxNC44eiIvPjwvZz48L3N2Zz4=");
}
.e-filemanager .e-large-icons .e-fe-json {
 background-image:
url("data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAOEAAADhCAMAAAAJbSJIAAAAkFBMVEXp6eD///+Xd6jZ18qSb6Xr6+HW0NOgg6/t7uOUc6aScKtm5tvs5+/Qw9fOxc2OaqG3o8OxnLnBr8qmi7S/sMPBtMT49vmZeqrMwsy2pL3LvdPl493DssypkbTl3umjibDZz9/d2di1oMHy7vSnjRn4evfluTXzN3TzNHGusjcl9e0obytl7f39/SkiLOGXZqgHBjVAAAKqULEQVR4nO2da3vqIBKANQtN4FRjfqzVeJ5dqq611//+/28QAgVwwofg4y3zp0SM82bIADNEO12Z/MDOrwj8kZqlJB3ZyV/iS+WP1K67Ef78GmDnz3/uhihT9HuAKeHdECV6ftGfGGe9ECVqfu8pzAnvhGgz4X0QJUP+EAZA3gXRbsJ7IFpOeAde2wlvR7Se8GZE+wlwRXSA8EZEfwhv3SC8CZENwhvQXSE8AZEvwj1EZ0h1EZ0h1AX0SFCTUSXCPUQnSLUQnSLUAfRMUINRNcI1RGdI1RGdI9QfDFBQkVEFwnVEJ0kVEJ0k1AF0VFCBURXCdsjOkvYgtFdwraIDhO2RHSZsB2i04StEN0mbIPoOGE LRNcJryM6T3gV0X3Ca4j/AOEVxH+BUI54b0IYpaJ1ZXyhpLuU1AiliHclhNH8OD0tp3P17VTRzro8TSfziFz6R00k+1HvSRh9LQFGqeAnVUR4AJcLwXKj1QPgQwjhN0ABEWUn0isRmEYaG+oeQ7jC1C8Aik6EfcCuxbuOBuIDCOGKOTC1sq9IOcnuTob2Gh3VPGE05WwM4o2igZuYuxqM1BGNE0LRxG9VE6MXwF8/V7z8AYTRiYUKDMBU/UGC0ziNw1SH8h0yTgjtI/iZf9rrhPwo/lxf0m1IOVgY5yQRgr8ls5L9LZPp9dFI6IGHGwjKakg+mEQV7Pit0oxSuNE37nhHhy2/b36Jg7EU2tI3xBWuNgWWCfEL54wrJ4Qk94TTxhW/GEzWIJYZbEkTdwmxBGH+lpe
```

```
pAxuk0IN2GWiWghJE/1NiFdfiDUvPpzmmjDasdXfstF8lwnhV5EEAF+NrVwmPBZ5HHZ8FwmjUZGKQ
42rP5cJ+XyhpJXDhJ051/NtHC8sJtyRGNKcXyKsQrh5AQ8P4Fq4bRDThJA8ZECSCYar3HqwarYeb
qinLSOknSsIJFlACEcBwCAYSaZtcI6udoZ6MUWYhQRQnmqL4FP/qyNvQpJtgctRtWKWkd1izQMdk
au51OiNqlKsXZgkhNGIPjxAo/hbUjanUx/wplRHNER4We8dziwXr5zlrAodd1Ivrg7RtfVkiWYIo
81o+pIU45xk0dBe5sXMACQv01HLqrcRwugYp+u9oDBocrsLM61cnS3VHx9baTVBCJ/4kmHaq9RLY
rUSvWBBb9y4EuHFBCGtVdDbfboPYBpvl6LiVlk+3EcKTYIhssFct2AnUb50RwqlIGN4NMRQJW42MJ
gj5qHex5MbaIZVoJd65doOsEUL4FGRboApTNLZQ1ACoihXbZJKWgUaQ+Mh7Bwmb6eCUbayaCvF6
iLle3mbHFr2fTOEWdyLovk3teke4TRigRRM59xmXwtiijA3ql/MSzW5CmH7ckDfhnpkpwGc2oW91
OVXnMISVilg68/PxCxh0bXkqzo4f5se5ZExOmt2eMOE8Ik6UWZ/2gojjGTbFtnOI7Cxa43fiVpEU
zgnjWSZDnqrQsvyNEWuTeIgSJ5WINlZAw9YMyobJ2xRe6JBpDnjbXW+1BN2PGEunrBZPKEnvCae8
NLGE3Y8oUw8oSe8Jq32tdG1hayN24R5KhsleJ1ue3bgJgEYJLLVu+OEqRcnbnXp/tNiwu/rUaTTI
tfGopF978zQJGCo8xIvpyfJ1dj33hPdyxSg09WUaKMOGHVowU71PeIHELKSH0pGR+VX6y4qDsdpw
kogypebJuSLiQiDs3p2f34uXiFtWTIUxDghv/UwtVB1R1MnCVmioTSxXC/mCYUtFC13FxTCMsH5U
6ixqcM8YdpPOSNveh8f6Gw8egChgBirfqcC58NYq5T8CMJONEH01Yqz9nOIA8WYgPGEHhai+ehS2
QehRizNXqhJZ60jzR0djyHMDvJ9HUCjnRfYIJyMRpNN6516ZXkQYSev7GvOaRSq9lV5HOFvisf0h
PaLJ/SE9osn9IT2iyf0hPaLJ/SE9osn9IT2iyf0hPaLJ/SE9osWIXRJtAifXBitwgs4I4kWofjCs
d0SekJPaL14Qk9ov3hCT2i/eEJP2CzoIsYBrooxwvPrRYbl4xjgcDgchkEMcN11AQJxkLY4JzGuv
T8IX6Si9yI1V5giRIO85aL0rUDBYk90jP/OznH1uvj0+U4afAxqbgJa9C5S5vi8HN1WEU0Tvgpbm
HFP1PNccjECr2OhwazSz8FzfYHhMNDcrR6Sx5JiM5juSJ8fi83GJ9EEkZYutQKQnSu6FkL5oNB3
WfNkX7MCD+E41YQBhUPinraA/2Hif2REXaFb6ezgRB80Ks/FoPBYLEuuRCdukWD7WnQYzdkwZtdE
L7zTrSBkh7Scxjjyy82gl7CX8EyRjOQ/6RjzDrtpK+SSUbYfeVwLCCkYfRvcecFa2Lq4hlzLE6IG
985X3OEY7sIiWXj+mE+QDuiesj/3CH1K/c0c4Q8jwWEKPFHrIEw/pu3nwlhBRMt4+IoT9gturkFh
EH+/7JhpkrsKLs4JjzFXEUgLEYM6wnxLD9ddjHaEhTmRIGwy34Zwx7CRX0vpb5KKify44VvRcJn6
kQLCEE+IXuvTrYzswSDixPr/MywFLG65M8AWUNIR4teHSJaNhmIyERnUSYkSNS5FhCyEX0Nqo8iW
uTnBtVTQ3pVifC/YvClgZAGk+54GZdBqIN3VUJyY/ZlwphanlhDyIa89BkqM9I5a41enI+j7Pl1h
IDclTzMWkEY4D1T8LwVGAGBr1EUk51lbhTBARNclCN1BGMSfhYr3LRdyAGGvGSwLovL/mAShSwC2h
DCIt9wacZ9UJiolcZaewtWmZAC6xCdbCAPE4k0m23L4UPIhW1Fms1ZrCLNsG5eNGtBvXdV5DoXpu
kWE2bKveBzJVEUnlvLWJ8gqwtTKkCVN836qMx5e/iH3ah1bRpj2LzKJIEclcxqSoKvMaXKf0hEjP
NtGyDKHzxfLNealOSEmt2YfWkfIxxZiKeGtDBE0H+7qCdmIsbCPkHa/XFXz+pCE0sr6kBDsJXLXQ
kLyCOWVi8Y1Phnyqmt86ul4zxtmI+FzSGNcqlLQEa+ap2H9WagVPJjwcY2QxHdynrpCzLXxs5YmQ
sBNds0SitVJmuCtif/EsLxf0pGcRtPumS9W0VIH192pmcI+iBD/HXAlWDykBrMG52EsGF8yia0ez
1zOmwTKd/5LeCqEdMQwTJiFivHnKY4BxhjErIzEPjHttvvXgFSuURy+lzUxS14vjbK6BV2I8CnIG
h/GxYLFICGru6xnsx6rMnU/ixh4OfjcGwzD8DyglbJkgWZxf9zbDofbovYkPJslhNyIYY4Q7+sbF
uMbrlYPxZILfTir81lbP+TnBsWIYY5wV9+O2zhQLQCnzIH+Li2RPpZXwMWjjL15ghR+FHTbMDd/
bDqW31ZC15WGw1KE7laQrA2Tpj9Xt+stM3gbyh8HJ6J5o/Ltmc3Ci/ERr2gbHJ9woMmY82O+BicF
x/UwOdZCCrnl709OT9eb2sSw1kjbvKU6xh+vqKyjiZDO+ozPadJxIhnudsMQxHUVGISz/U7nYYjie
r68URxkOpKGNqh2O1n90fsTkg+TnW0nd9wQ5/cmekL7xRP+vxKe7hfqTAu3mUyFsF8diW0V0Nci7
K5ccsJaSShkhD+r5kmJRYLA6keTsNtdnyq5TeskOa21DP8DUTgufjy2Bo0AAAAASUVORK5CYII="
);
}
</style>
```

## HOMECONTROLLER MVC.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
```



```

{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder

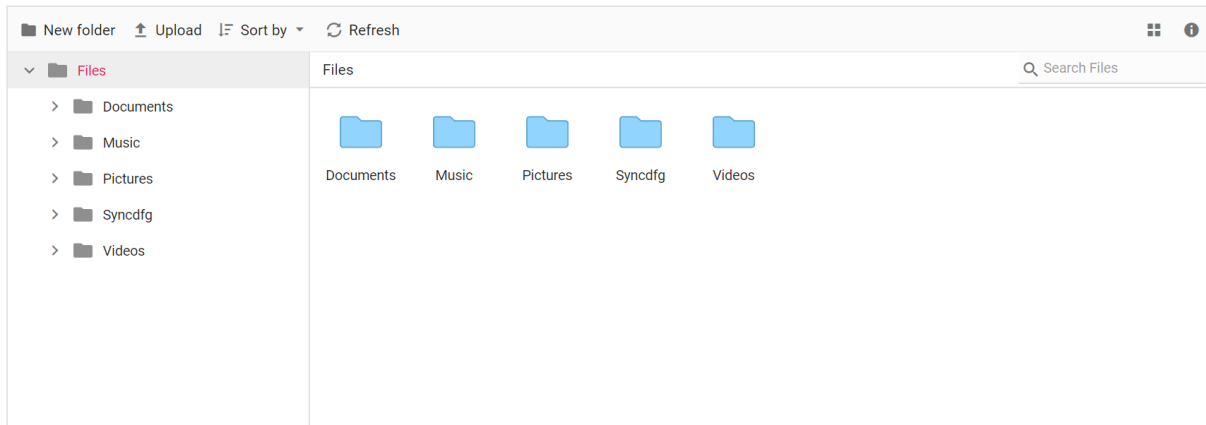
```

```

 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Output be like the below.



### Nested FileManager

FileManager can be rendered inside the other components like Tab, Dialog, and more.

- [Adding file manager inside the dialog](#)
- [Adding file manager inside the tab](#)

#### *Adding file manager inside the dialog*

The following example shows the file manager component rendered inside the dialog. Click the browse button in the Uploader element to open the File Manager inside the Dialog control.

#### **CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div class="control-section">
 <div class="sample-container">
 <div id='container' class="fileupload">
 @Html.EJS().Uploader("UploadFile").Render()
 @Html.EJS().Button("openBtn").Content("Browse...").Render()
 </div>
 <div id='target' class="control-section">
 @*Dialog*@
 @Html.EJS().Dialog("dialog").AnimationSettings(new
DialogAnimationSettings() { Effect = DialogEffect.None
}).Created("onDialogLoad").Close("dialogClose").Open("dialogOpen").Header("O
pen").Visible(false).ShowCloseIcon(true).Width("850px").Target("#target").Co
ntentTemplate(@<div class='dialogContent'>
 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).FileOpen("onFileOpen").Render()
 </div>).Render()
 </div>
 </div>
</div>
</script>
```

```

function onDialogLoad() {
 document.getElementById("openBtn").addEventListener('click',
function () {
 var dialogObj =
document.getElementById('dialog').ej2_instances[0];
 dialogObj.show();
 var filemanagerObj =
document.getElementById('filemanager').ej2_instances[0];
 filemanagerObj.path = "/";
 filemanagerObj.refresh();
 });
}
function dialogOpen() {
 document.getElementById('container').style.display = 'none';
}
function dialogClose() {
 document.getElementById('container').style.display = 'block';
}
function onFileOpen(args) {
 var file = args.fileDetails;
 if (file.isFile) {
 args.cancel = true;
 var uploadObject =
document.getElementById('UploadFile').ej2_instances[0];
 var dialogObj =
document.getElementById('dialog').ej2_instances[0];
 uploadObject.files = [{ name: file.name, size: file.size, type:
file.type }];
 dialogObj.hide();
 }
}
</script>
<style>
 .fileupload {
 max-width: 500px;
 margin: auto;
 }
 #openBtn {
 position: absolute;
 top: 67px;
 margin-left: 13px;
 }
 #target {
 height: 550px;
 }
 #dialog {
 top: 20px !important;
 max-height: 500px !important;
 left: 20px !important;
 }
</style>

```

**HOMECONTROLLER CORE.CS**

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// use the package for hosting
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Http.Features;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 public PhysicalFileProvider operation;
 public string basePath;
 // Root Path in which files and folders are available.
 string root = "wwwroot\\Files";
 public HomeController(IHostingEnvironment hostingEnvironment)
 {
 // Map the path of the files to be accessed with the host
 this.basePath = hostingEnvironment.ContentRootPath;
 this.operation = new PhysicalFileProvider();
 // Assign the mapped path as root folder
 this.operation.RootFolder(this.basePath + "\\\" + this.root);
 }
 public object FileOperations([FromBody] FileManagerDirectoryContent
args)
 {
 // Restricting modification of the root folder
 if (args.Action == "delete" || args.Action == "rename")
 {
 if ((args.TargetPath == null) && (args.Path == ""))
 {
 FileManagerResponse response = new
FileManagerResponse();
 ErrorDetails er = new ErrorDetails
 {
 Code = "401",
 Message = "Restricted to modify the root folder."
 };
 response.Error = er;
 return this.operation.ToCamelCase(response);
 }
 }
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
to show/hide hidden items
 return
this.operation.ToCamelCase(this.operation.GetFiles(args.Path,
args.ShowHiddenItems));
 case "delete":

```

```

 // Path - Current path where of the folder to be
 deleted; Names - Name of the files to be deleted
 return
 this.operation.ToCamelCase(this.operation.Delete(args.Path, args.Names));
 case "copy":
 // Path - Path from where the file was copied;
 TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 with same name in the copied location that is confirmed for renaming;
 TargetData - Data of the copied file
 return
 this.operation.ToCamelCase(this.operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 Path where the file/folder is to be moved; RenameFiles - Files with same
 name in the moved location that is confirmed for renaming; TargetData - Data
 of the moved file
 return
 this.operation.ToCamelCase(this.operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData));
 case "details":
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 this.operation.ToCamelCase(this.operation.Details(args.Path, args.Names));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 this.operation.ToCamelCase(this.operation.Create(args.Path, args.Name));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 this.operation.ToCamelCase(this.operation.Search(args.Path,
 args.SearchString, args.ShowHiddenItems, args.CaseSensitive));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 this.operation.ToCamelCase(this.operation.Rename(args.Path, args.Name,
 args.NewName));
 }
 return null;
}

// Processing the Upload operation
public IActionResult Upload(string path, IList<IFormFile>
uploadFiles, string action)
{
 // Here we have restricted the upload operation for our online
 samples
 if (Response.HttpContext.Request.Host.Value ==
 "ej2.syncfusion.com")
 {
 Response.Clear();
 }
}

```

```

 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 403;

Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
"File Manager's upload functionality is restricted in the online demo. If
you need to test upload functionality, please install Syncfusion Essential
Studio on your machine and run the demo";
 }
 // Use below code for performing upload operation
 else
 {
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 }
 return Content("");
}
// Processing the Download operation
public IActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public IActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return this.operation.GetImage(args.Path, args.Id, false, null,
null);
}
public IActionResult Index()
{
 return View();
}
}
}

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;

```

```

// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }
 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file
 return
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 // requested; Name - Names of the requested folders
 return
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 }
 }
 }
}

```

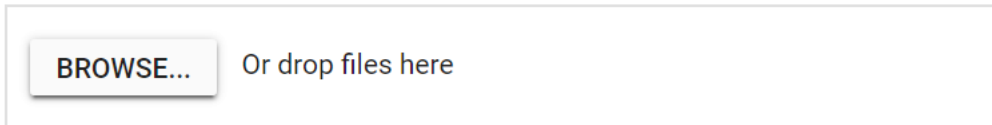


```

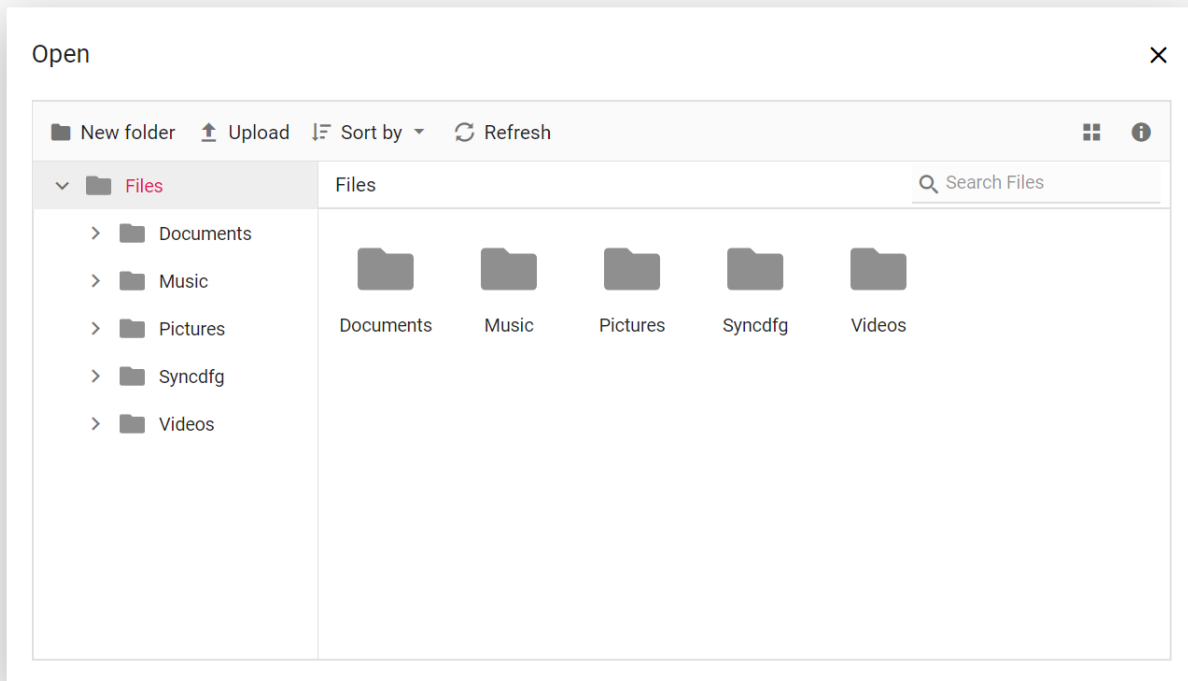
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 JsonResult(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 JsonResult(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
 args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 JsonResult(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
 args.NewName)));
 }
 return null;
}
// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
 null);
 return Content("");
}
// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
 JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}
// Processing the GetImage operation
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
 null);
}
public ActionResult Index()
{
 return View();
}
}
}

```

Initially output be like the below.



After clicking the file browser button, file manager is rendered with dialog. Output be like the below.



#### *Adding file manager inside the tab*

The following example demonstrate that the file manager component is placed inside the content area of tab element.

#### **CSHTML**

```
@using Syncfusion.EJ2.Navigations;
<div class="e-tab-section">
 <div class="col-lg-8 content-wrapper control-section">
 <div class="e-sample-resize-container">
 <div id="overview" style="display: none">
 <div class="content-title">
 <div class="cnt-text">Overview</div>
 </div>
 The file manager component contains a context menu for
 performing file operations, large-icons view for displaying the files and
 folders, and a breadcrumb for navigation. However, these basic
 functionalities can be extended by using the additional feature modules like
 toolbar, navigation pane, and details view to simplify the navigation and
 file operations within the file system.
 </div>
 </div>
 </div>
</div>
```

```

 </div>
 <div id="filemanager" style="display: none">
 <div class="content-title">
 <div class="cnt-text">Filemanager with default
functionalities</div>
 </div>
 <div>
 @Html.EJS().FileManager("file").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).Render()
 </div>
 </div>
 @ (Html.EJS().Tab("orientationTab")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText1, Content =
"#overview" },
 new TabBarItem { Header = ViewBag.headerText2, Content =
"#filemanager" },
 }).Height("320").ShowCloseButton(true).Selected("onSelect").Render()
)
 </div>
</div>
<script>
 function onSelect() {
 var fileObj = document.getElementById("file").ej2_instances[0];
 fileObj.refreshLayout();
 }
</script>
<style>
 .e-content .e-item {
 font-size: 12px;
 padding: 10px;
 text-align: justify;
 }
 .content-title {
 height: 50px;
 display: table;
 margin: 0 auto;
 }
 .cnt-text {
 vertical-align: middle;
 display: table-cell;
 font-size: 18px;
 font-weight: 600;
 }
</style>

```

### HOMECONTROLLER MVC.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Navigations;
//File Manager's base functions are available in the below package
using Syncfusion.EJ2.FileManager.Base;
//File Manager's operations are available in the below package
using Syncfusion.EJ2.FileManager.PhysicalFileProvider;
using Newtonsoft.Json;
// Use the package for hosting
using System.Web.Hosting;
namespace WebApplication.Controllers
{
 public class HomeController : Controller
 {
 // Accessing the File Operations from File Manager package
 PhysicalFileProvider operation = new PhysicalFileProvider();
 public HomeController()
 {
 // Map the path of the files to be accessed with the host
 var path = HostingEnvironment.MapPath("~/Content/Files");
 // Assign the mapped path as root folder
 operation.RootFolder(path);
 }

 public ActionResult FileOperations(FileManagerDirectoryContent args)
 {
 // Processing the File Manager operations
 switch (args.Action)
 {
 case "read":
 // Path - Current path; ShowHiddenItems - Boolean value
 // to show/hide hidden items
 return
 Json(operation.ToCamelCase(operation.GetFiles(args.Path,
 args.ShowHiddenItems)));
 case "delete":
 // Path - Current path where of the folder to be
 // deleted; Names - Name of the files to be deleted
 return
 Json(operation.ToCamelCase(operation.Delete(args.Path, args.Names)));
 case "copy":
 // Path - Path from where the file was copied;
 // TargetPath - Path where the file/folder is to be copied; RenameFiles - Files
 // with same name in the copied location that is confirmed for renaming;
 // TargetData - Data of the copied file
 return
 Json(operation.ToCamelCase(operation.Copy(args.Path, args.TargetPath,
 args.Names, args.RenameFiles, args.TargetData)));
 case "move":
 // Path - Path from where the file was cut; TargetPath -
 // Path where the file/folder is to be moved; RenameFiles - Files with same
 // name in the moved location that is confirmed for renaming; TargetData - Data
 // of the moved file

```

```

 return
 }
 Json(operation.ToCamelCase(operation.Move(args.Path, args.TargetPath,
args.Names, args.RenameFiles, args.TargetData)));
 case "details":
 if (args.Names == null)
 {
 args.Names = new string[] { };
 }
 // Path - Current path where details of file/folder is
 requested; Name - Names of the requested folders
 return
 }
 Json(operation.ToCamelCase(operation.Details(args.Path, args.Names)));
 case "create":
 // Path - Current path where the folder is to be
 created; Name - Name of the new folder
 return
 }
 Json(operation.ToCamelCase(operation.Create(args.Path, args.Name)));
 case "search":
 // Path - Current path where the search is performed;
 SearchString - String typed in the searchbox; CaseSensitive - Boolean value
 which specifies whether the search must be casesensitive
 return
 }
 Json(operation.ToCamelCase(operation.Search(args.Path, args.SearchString,
args.ShowHiddenItems, args.CaseSensitive)));
 case "rename":
 // Path - Current path of the renamed file; Name - Old
 file name; NewName - New file name
 return
 }
 Json(operation.ToCamelCase(operation.Rename(args.Path, args.Name,
args.NewName)));
 }
 return null;
}

// Processing the Upload operation
public ActionResult Upload(string path,
IList<System.Web.HttpPostedFileBase> uploadFiles, string action)
{
 FileManagerResponse uploadResponse;
 //Invoking upload operation with the required paramaters
 // path - Current path where the file is to uploaded;
 uploadFiles - Files to be uploaded; action - name of the operation(upload)
 uploadResponse = operation.Upload(path, uploadFiles, action,
null);
 return Content("");
}

// Processing the Download operation
public ActionResult Download(string downloadInput)
{
 FileManagerDirectoryContent args =
JsonConvert.DeserializeObject<FileManagerDirectoryContent>(downloadInput);
 //Invoking download operation with the required paramaters
 // path - Current path where the file is downloaded; Names -
 Files to be downloaded;
 return operation.Download(args.Path, args.Names);
}

// Processing the GetImage operation

```

```
public ActionResult GetImage(FileManagerDirectoryContent args)
{
 //Invoking GetImage operation with the required paramaters
 // path - Current path of the image file; Id - Image file id;
 return operation.GetImage(args.Path, args.Id, false, null,
null);
}
public ActionResult Index()
{
 List<TabTabItem> orientationItems = new List<TabTabItem>
 {
 new TabTabItem { Header = new TabHeader { Text = "Overview"
}, Content = "#overview" },
 new TabTabItem { Header = new TabHeader { Text =
"Filemanager" }, Content = "#filemanager" }
 };
 ViewBag.orientationItems = orientationItems;
 return View();
}
}
```

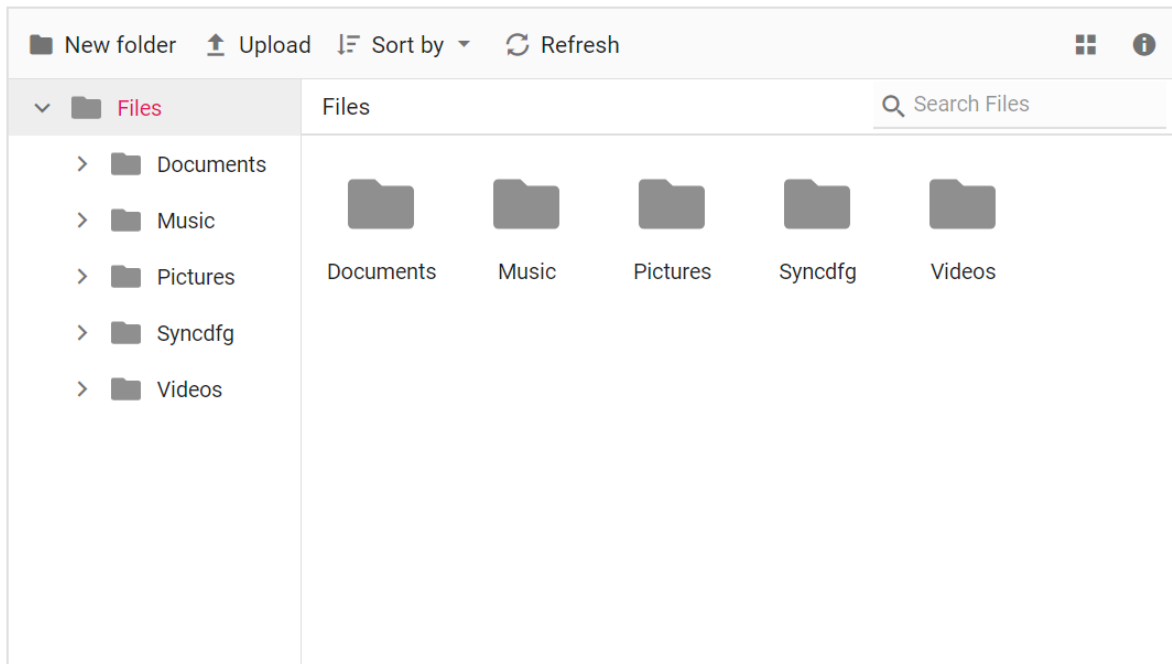
Output be like the below for initial view.

OVERVIEW × FILEMANAGER ×

## Overview

The file manager component contains a context menu for performing file operations, large-icons view for displaying the files and folders, and a breadcrumb for navigation. However, these basic functionalities can be extended by using the additional feature modules like toolbar, navigation pane, and details view to simplify the navigation and file operations within the file system.

Output be like the below, when file manager is placed inside the tab

OVERVIEW × **FILEMANAGER** ×**Filemanager with Default Functionalities****Render FileManager in Internet Explorer**

In the Internet Explorer browser, ES6 promise polyfills are required to run the file manager component. Click this [link](#) to learn more about the EJ2 components browser compatibility.

To render the file manager component in Internet Explorer browser, add the ES6 promise polyfill script above the Syncfusion Essential JS 2 Scripts in **layout** page.

```
`html
<head>
<!-- ES6 Promise polyfill -->
<script src="https://cdn.polyfill.io/v2/polyfill.min.js"></script>
<!-- Syncfusion Essential JS 2 Scripts -->
<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
</head>
`
```

After adding the script file in **layout** page, render the file manager component. The following example shows how to render the file manager in IE browser.

**CSHTML**

```
<div class="control-section">
 <div class="sample-container">
 <!-- Declare filemanager element -->
```

```

 @Html.EJS().FileManager("filemanager").AjaxSettings(new
Syncfusion.EJ2.FileManager.FileManagerAjaxSettings
 {
 Url = "/Home/FileOperations",
 GetImageUrl = "/Home/GetImage",
 UploadUrl = "/Home/Upload",
 DownloadUrl = "/Home/Download"
 }).Render()
 <!-- end of filemanager element -->
</div>
</div>

```

## LAYOUT MVC.CSHTML

```

<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>@ViewBag.Title - My ASP.NET Application</title>
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css" />
 <!-- ES6 Promise polyfill -->
 <script src="https://cdn.polyfill.io/v2/polyfill.min.js"></script>
 <!-- Syncfusion Essential JS 2 Scripts -->
 <script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
 @Styles.Render("~/Content/css")
 @Scripts.Render("~/bundles/modernizr")
</head>
<body>
 <div class="navbar navbar-inverse navbar-fixed-top">
 <div class="container">
 <div class="navbar-header">
 <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">

 </button>
 @Html.ActionLink("Application name", "Index", "Home", new {
area = "" }, new { @class = "navbar-brand" })
 </div>
 <div class="navbar-collapse collapse">
 <ul class="nav navbar-nav">
 @Html.ActionLink("Home", "Index", "Home")
 @Html.ActionLink("About", "About", "Home")
 @Html.ActionLink("Contact", "Contact", "Home")

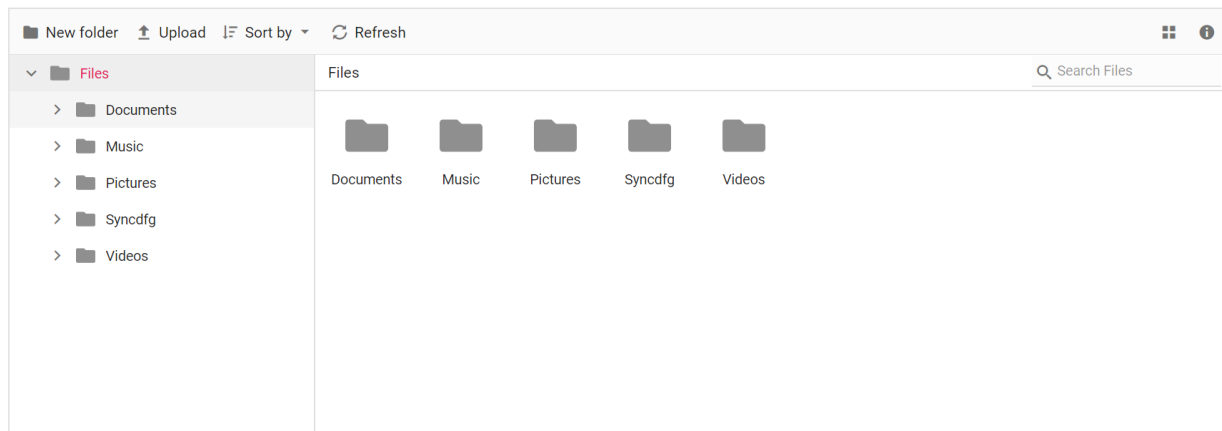
 </div>
 </div>
 </div>
 <div class="container body-content">
 @RenderBody()
 <hr />
 <footer>

```



```
<p>© @DateTime.Now.Year - My ASP.NET Application</p>
</footer>
</div>
@Html.EJS().ScriptManager()
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>
```

Output be like the below.



### Create the custom file provider using NodeJS

Here we manipulate the Azure Blob Storage to supply the necessary data for the File Manager. We achieve this by utilizing NodeJS to fetch the required data from the Azure blob storage.

NodeJS acts as the bridge between the File Manager component and Azure Blob Storage, allowing seamless communication and data retrieval. Through this integration, the File Manager can access and interact with the data stored in Azure Blob Storage, enabling smooth file management operations.

#### Prerequisites

- Valid Azure blob storage account. ( accountName, accountKey, endpointSuffix)
- Node version 14 above.

#### Introduction to Azure Blob Storage

Azure Blob Storage is a cloud-based object storage service provided by Microsoft Azure. It is designed to store and manage unstructured data, also known as "blobs" in the cloud. Blobs can be any type of data, such as images, videos, documents, backups, logs, and more.

#### Key concepts of Azure Blob Storage

**Containers:** In Azure Blob Storage, data is organized into containers. Containers are logical units that can hold one or more blobs. Think of them as directories or folders that help organize the data.

**Blobs:** Blobs are the actual data objects stored in Azure Blob Storage.

By understanding the fundamental concepts and use cases of Azure Blob Storage, you will be well-prepared to proceed with setting up and interacting with it using NodeJS in the custom File Provider.

### Create NodeJS project

Following the steps to create the NodeJS project.

Create a new directory for your project and run the following command to initialize a new NodeJS project. This will create a package.json file.

```
`ts
npm init
`
```

Install the following packages.

- express
- @azure/storage-blob
- archiver
- body-parser
- cors
- esm
- multer

Open your text editor or integrated development environment (IDE) and create the index.js file start writing your NodeJS code. This file will serve as the entry point of your application.

```
`ts
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
res.send('Hello, NodeJS!');
});
app.listen(port, () => {
console.log(Server running on http://localhost:${port});
});
`
```

To start your NodeJS application, simply run the following command in your terminal, pointing to the entry point file:

```
`ts
node index.js
`
```

### Initialize container client

We need to first get the BlobServiceClient. By using the connection string, we can obtain the BlobServiceClient. So, format the connection string as shown below.

```
`ts
```

```
Const connectionString =
DefaultEndpointsProtocol=https;AccountName=${accountName};AccountKey=${accountKey};E
ndpointSuffix=${EndpointSuffix};
`
```

We can obtain the `BlobServiceClient` and the **containerClient** using this connection String and the `BlobServiceClient`. the **containerName** is the container from your Azure blob storage account that you need to access.

```
`ts
```

```
import { BlobServiceClient } from "@azure/storage-blob";
const blobServiceClient = BlobServiceClient.fromConnectionString(connectionString);
const containerClient = blobServiceClient.getContainerClient(containerName);
`
```

#### File actions

Need to provide the following action to creating a new folder, copying and moving of files or folders, deleting, uploading, and downloading the files or folders in the file system

#### Read

Specify the directory name that needs to be accessed.

```
`ts
```

```
const directoryName = 'Files';
`
```

Create the **app.post** method with URL **‘/fileManager’**.

To identify the action by use this condition **req.body.action === ‘read’**

The following table represents the request parameters of **read** operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	read	Name of the file operation.
path	String	-	Relative path from which the data has to be read.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details about the current path (directory).

*Example for request:*

```
`ts
```

```
{
 action: "read",
 path: "/Videos/",
}
```

```
showHiddenItems: false,
data: [
0:{
name:"Videos",
size:0,
dateModified:"2023-09-14T14:28:27.000Z",
dateCreated: "2023-09-14T11:16:57.000Z",
hasChild:true,
isFile:false,
type:"Directory",
filterPath:"/",
fmicon: "e-fe-folder",
fmiconClass: "e-fe-folder",
fmid: "fetree0",
fmmodified: "September 14, 2023 19:58"
}
]
}
,
```

The following table represents the response parameters of **read** operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	<a href="#">FileManagerDirectoryContent</a>	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of files and folders present in given path or directory.
error	<a href="#">ErrorDetails</a>	-	Error Details

<a id="file-manager-directory-content"></a>

The following table represents the contents of **FileManagerDirectoryContent** in the file manager request and response.

Parameter	Type	Default	Explanation	Is required
----	----	----	----	----
name	String	-	File name	Yes
dateCreated	String	-	Date in which file was created (UTC Date string).	Yes
dateModified	String	-	Date in which file was last modified (UTC Date string).	Yes

filterPath	String	-	Relative path to the file or folder.	Yes
hasChild	Boolean	-	Defines this folder has any child folder or not.	Yes
isFile	Boolean	-	Say whether the item is file or folder.	Yes
size	Number	-	File size	Yes
type	String	-	File extension	Yes
permission	[AccessRules](#)	-	File extension	Optional
caseSensitive	Boolean	-	Defines search is case sensitive or not.	Optional
action	String	read	Name of the file operation.	Optional
names	String[]	-	Name list of the items to be downloaded.	Optional
data	FileManagerDirectoryContent	-	Details of the download item.	Optional
uploadFiles	IList<IFormFile>	-	File that are uploaded.	Optional
newName	String	-	New name for the item.	Optional
searchString	String	-	String to be searched in the directory.	Optional
targetPath	String	-	Relative path where the items to be pasted are located.	Optional
targetData	FileManagerDirectoryContent	-	Details of the copied item.	Optional
renameFiles	String[]	-	Details of the renamed item.	Optional
<a id="access-rules"></a>

The following table represents the **AccessRules** properties available for file and folder:

Properties	Applicable for file	Applicable for folder	Description
---	---	---	---
Copy	Yes	Yes	Allows access to copy a file or folder.
Read	Yes	Yes	Allows access to read a file or folder.
Write	Yes	Yes	Allows permission to write a file or folder.
WriteContents	No	Yes	Allows permission to write the content of folder.
Download	Yes	Yes	Allows permission to download a file or folder.
Upload	No	Yes	Allows permission to upload to the folder.
Path	Yes	Yes	Specifies the path to apply the rules, which are defined.
Role	Yes	Yes	Specifies the role to which the rule is applied.
IsFile	Yes	Yes	Specifies whether the rule is specified for folder or file.

*Example for response:*

```
`ts
{
 cwd:
```

```
{
 filterPath: "/",
 hasChild: true,
 name: "Videos",
 size: 0,
 type: "File Folder"
},
files:[
 0:{
 dateCreated: "2023-09-14T11:16:57.000Z"
 dateModified: "2023-09-14T11:16:57.000Z"
 filterPath: "/Videos/"
 hasChild: false
 isFile: true
 name: "about.txt"
 size: 29
 type: ".txt"
 }
],
error:null
}
```

#### Get image

Create the **app.get** method with URL **'/fileManager/GetImage'**.

The following table represents the request parameters of **GetImage** operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------

path	String	-	Relative path to the image file
------	--------	---	---------------------------------

The req.query.path contains the exact path of the images. For example: **"/Jack.png"**.

Download the blob (image) from Azure Blob Storage using the blobClient and stores the result in the downloadResponse variable.

Pipe the readableStreamBody from the blob to the res response. It means the image data will be streamed from the Azure Blob Storage directly to the client's browser when the image URL is accessed.

Handle the exception if the image is not available in the given path.

### Download

Create the **app.post** method with URL **‘/fileManager/Download’**.

The following table represents the request parameters of *download* operations.

Parameter	Type	Default	Explanation
action	String	download	Name of the file operation
path	String	-	Relative path to location where the files to download are present.
names	String[]	-	Name list of the items to be downloaded.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details of the download item.

*Example for request:*

```
`ts
{
 action: 'download',
 path: '/Downloads/Testing/',
 names: ['About.txt'],
 data: [
 0:{
 name: 'About.txt',
 type: '.txt',
 isFile: true,
 size: 29,
 dateModified: '2023-09-14T06:03:52.000Z',
 hasChild: false,
 filterPath: '/Downloads/Testing/',
 fmcreated: null,
 fmmodified: 'September 14, 2023 11:33',
 fmiconClass: 'e-fe-txt',
 fmicon: 'e-fe-txt'
 }
]
}
```

The **req.body.downloadInput** must be parsed to get the **downloadObj**. Download the blob from Azure Blob Storage using the blobClient.

Download the blob from Azure Blob Storage using the blobClient and Pipe the readableStreamBody to the response object.

Create the archive file to download the multiple Files, Folders and single folders, then pipe the archive to the response.

### Upload

Create the **app.post** method with URL **'/fileManager/Upload'**.

The following table represents the request parameters of *Upload* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	Save	Name of the file operation.
path	String	-	Relative path to the location where the file has to be uploaded.
uploadFiles	IList<IFormFile>	-	File that are uploaded.

*Example for request:*

```
`ts
{
 path: '/Pictures/',
 action: 'save',
 data: [
 0:{
 name: 'Pictures',
 type: 'File Folder',
 isFile: true,
 size: 0,
 dateModified: '2023-09-14T06:03:52.000Z',
 hasChild: true,
 filterPath: '',
 fmid: 'fetree1',
 }
],
 filename: 'bird (2).jpg'
}
```

Multer is a popular middleware used to handle file uploads in Express-based web applications. Create the Multer config to store the upload files in buffer.



```
`ts
const multerConfig = {
storage: memoryStorage()
};
`
```

Need to handle the 3 cases here.

- Save
- Keep Both (action name will be **keepboth**)
- Replace (action name will be **replace**)

Create the **getBlockBlobClient** with the **req.body.filename**. If the blob does not exist, then upload the data to that blob. If the blob already exists, then create an error message containing "File Already Exists" and send the response.

#### Create a new folder

The following table represents the request parameters of *create* operations.

Parameter	Type	Default	Explanation
action	String	create	Name of the file operation.
path	String	-	Relative path in which the folder has to be created.
name	String	-	Name of the folder to be created.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details about the current path (directory).

*Example for request:*

```
`ts
action: "create",
data: [
0:{
filterPath: "/",
hasChild: true,
isFile: false,
name: "files",
nodeId: "fe_tree",
size: 0,
type: ""
}
],
```

```
name: "Hello",
path: "/test/"
,
```

Check the existence of the folder, If the folder exists then send the error message containing “Folder already exists”. If it does not exist, then create the folder. Create the folder by creating the file in that folder’s path.

The following table represents the response parameters of *create* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the created folder
error	<a href="#">ErrorDetails</a>	-	Error Details

*Example for response:*

```
`ts
{
 cwd: null,
 files: [
 0:{
 dateCreated: "2023-09-14T10:52:25.000Z",
 dateModified: "2023-09-14T10:52:25.000Z",
 filterPath: null,
 hasChild: false,
 isFile: false,
 name: "New",
 size: 0,
 type: "Directory"
 }
],
 details: null,
 error: null
}
```

### [Rename](#)

The following table represents the request parameters of *rename* operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------

action	String	rename	Name of the file operation.	
path	String	-	Relative path in which the item is located.	
name	String	-	Current name of the item to be renamed.	
newName	String	-	New name for the item.	
data	<a href="#">FileManagerDirectoryContent</a>	-	Details of the item to be renamed.	

*Example for request:*

```
`ts
{
 action: "rename",
 data: [
 0:{
 dateCreated: "2023-09-14T10:41:17.000Z",
 filterPath: "/Pictures/Nature/",
 hasChild: false,
 iconClass: "e-fe-image",
 isFile: true,
 name: "seaviews.jpg",
 size: 95866,
 type: ".jpg"
 }
],
 newName: "seaview.jpg",
 name: "seaviews.jpg",
 path: "/Pictures/Nature/"
}
```

Renaming can be done by copy the folder or file from the source blob instance to target blob instance. If the file exists, then send the error message as response.

The following table represents the response parameters of *rename* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the renamed item.

|error| [ErrorDetails](#) | - |Error Details|

*Example for response:*

```
`ts
{
 cwd:null,
 files:[
 0:{
 name:"seaview.jpg",
 size:95866,
 dateModified:"2023-09-14T11:16:57.000Z",
 dateCreated:"2023-09-14T10:41:17.000Z",
 hasChild:false,
 isFile:true,
 type:".jpg",
 filterPath:"/Pictures/Nature/"
 }
],
 error:null,
 details:null
}
```

### Delete

The following table represents the request parameters of *delete* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	delete	Name of the file operation.
path	String	-	Relative path where the items to be deleted are located.
names	String[]	-	List of the items to be deleted.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details of the item to be deleted.

*Example for request:*

```
`ts
{
 action: "delete",
```

```
path: "/",
names: ["bird.jpg"],
data: [
 0:{
 dateModified: "2023-09-14T09:12:53.000Z",
 filterPath: "/",
 hasChild: false,
 iconClass: "e-fe-image",
 isFile: true,
 name: "bird.jpg",
 size: 102182,
 type: ".jpg"
 }
]
}
```

To delete the file, directly get the file instance and delete the file. To delete the folder, we need to get all files inside that folder and delete all those files.

Handle the null exception if file or folder is not available.

The following table represents the response parameters of *delete* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details about the deleted item(s).
error	<a href="#">ErrorDetails</a>	-	Error Details

*Example for response:*

```
`ts
{
 cwd: null,
 details: null,
 error: null,
 files: [
 0:{
 dateModified: "2023-09-14T09:12:53.000Z",
```

```
filterPath: "/",
hasChild: false,
iconClass: "e-fe-image",
isFile: true,
name: "bird.jpg",
size: 102182,
type: ".jpg"
}
]
}
,
```

### Details

The following table represents the request parameters of *details* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	details	Name of the file operation.
path	String	-	Relative path where the items are located.
names	String[]	-	List of the items to get details.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details of the selected item.

*Example:*

```
`ts
{
 action: "details",
 path: "/FileContents/",
 names: ["bird.jpg"],
 data: [
 0:{
 dateModified: "2023-09-14T09:12:53.000Z",
 filterPath: "/",
 hasChild: false,
 iconClass: "e-fe-image",
 isFile: true,
 name: "bird.jpg",
```

```
size: 102182,
type: ".jpg"
}
]
}
,
```

To get the file and folder details, iterate the **req.body.names** to get the details of files and folders. If the data is file, then get the file instance and get the properties using the **getProperties** method. If the data is Folder, then get the blobs details under that folder using **listBlobsFlat** method. Get the required properties and send final response. Handled the null exception if the file or folder is not available.

The following table represents the response parameters of *details* operations.

Parameter	Type	Default	Explanation
details	<a href="#">FileManagerDirectoryContent</a>	-	Details of the requested item(s).
error	<a href="#">ErrorDetails</a>	-	Error Details

*Example:*

```
`ts
{
 cwd:null,
 files:null,
 error:null,
 details:
 {
 created: "2023-09-15T06:04:12.000Z"
 isFile: true
 location: "Files/bird.jpg"
 modified: "2023-09-15T06:04:12.000Z"
 multipleFiles: false
 name: "bird.jpg"
 size: "100.0 KB"
 }
}
,
```

### Search

The following table represents the request parameters of *search* operations.

Parameter	Type	Default	Explanation
	----	----	----
action	String	search	Name of the file operation.
path	String	-	Relative path to the directory where the files should be searched.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
caseSensitive	Boolean	-	Defines search is case sensitive or not.
searchString	String	-	String to be searched in the directory.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details of the searched item.

*Example for request:*

```
`ts
{
 action: "search",
 path: "/asia/",
 searchString: "nature",
 showHiddenItems: false,
 caseSensitive: false,
 data: [
 0:{
 filterPath: "/",
 hasChild: true,
 name: "asia",
 size: 0,
 type: "File Folder",
 fmid: "fetree1"
 }
]
}
```

Replace the '\*' in the **req.body.searchString** and assign the result to new variable. Get all blobs under this directory and check that path contains the search string

The following table represents the response parameters of *search* operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------



|----|----|----|----|

|cwd|[FileManagerDirectoryContent](#)|-|Path (Current Working Directory) details.|

|files|FileManagerDirectoryContent[]|-|Files and folders in the searched directory that matches the search input.|

|error|[ErrorDetails](#)|-|Error Details|

*Example for response:*

```
`ts
{
 cwd:
 {
 name:"asia",
 size:0,
 dateModified:"2023-09-14T14:28:27.000Z",
 dateCreated:"2023-09-14T11:16:57.000Z",
 hasChild:true,
 isFile:false,
 type:"File Folder",
 filterPath:"/"
 },
 files:[
 0: {
 dateModified: "2023-09-15T06:22:00.000Z",
 filterPath: "/asia/",
 hasChild: false,
 isFile: true,
 name: "about.txt",
 size: 42,
 type: ".txt"
 }
],
 error:null,
 details:null
}
```

### Copy and move

The following table represents the request parameters of *copy* operations.

Parameter	Type	Default	Explanation
action	String	copy	Name of the file operation.
path	String	-	Relative path to the directory where the files should be copied.
names	String[]	-	List of files to be copied.
targetPath	String	-	Relative path where the items to be pasted are located.
data	<a href="#">FileManagerDirectoryContent</a>	-	Details of the copied item.
targetData	<a href="#">FileManagerDirectoryContent</a>	-	Details of the copied item.
renameFiles	String[]	-	Details of the renamed item.

*Example for request:*

```
`ts
{
 action: "copy",
 path: "/",
 names: ["bird.jpg"],
 renameFiles: [],
 targetPath: "/asia/",
 targetData: {
 filterPath: "/",
 hasChild: true,
 name: "asia",
 size: 0,
 type: "File Folder",
 fmid: "fetree1",
 },
 data: [
 0:{
 dateCreated: "2023-09-15T06:04:12.000Z",
 dateModified: "2023-09-15T06:04:12.000Z",
 filterPath: "/",
```

```
hasChild: false,
isFile: true,
name: "bird.jpg",
size: 102182,
type: ".jpg",
fmcreated: "September 15, 2023 11:34",
fmhtmlAttr: {class: "e-large-icon", title: "bird.jpg"},
fmiconClass: "e-fe-image",
fmimageAttr: {alt: "bird.jpg"},
fmimageUrl: "http://localhost:3000/GetImage?path=%2Fbird.jpg&time=1694760243307",
fmmodified: "September 15, 2023 11:34",
}
]
}
,
```

Action name will be **move** for move action.

The following table represents the response parameters of *copy* operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	<a href="#">FileManagerDirectoryContent</a>	-	Path (Current Working Directory) details.
files	<a href="#">FileManagerDirectoryContent[]</a>	-	Details of copied files or folders
error	<a href="#">ErrorDetails</a>	-	Error Details

*Example for response:*

```
`ts
{
 cwd:null,
 files:[
 0:{
 dateCreated: "2023-09-15T06:55:03.000Z"
 dateModified: "2023-09-15T06:55:03.000Z"
 filterPath: "/asia/"
 hasChild: false
 isFile: true
```

```
name: "bird.jpg"
previousName: null
size: 102182
type: ".jpg"
}
],
error:null,
details:null
}
`
```

Need to handle two cases.

- Directory copy and move.
- File copy and move.

Create the **isRename** variable to store the is request is rename or not. If the **isRename** is false then check the existence of the folders, and if folder is existing, then send the error message. If **isRename** is true, then don't check the existence of the folder.

To move or copy the folders you need to get all the blobs from that folder and create the new path for each blob and copy the data from the old path to the new path. To move or copy the files copy the data from the source blob client to target client. If the action is move then delete the old blob.

---

**Note:** To get the complete project, refer to this [link](#)

---

## Floating Action Button

### Getting Started with ASP.NET MVC Floating Action Button Control

This section briefly explains about how to include [ASP.NET MVC Floating Action Button] control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

##### ~/ \_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

##### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

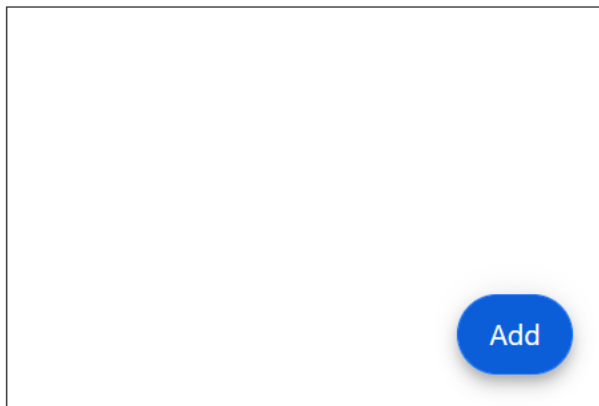
### Add ASP.NET MVC Floating Action Button control

Now, add the Syncfusion ASP.NET MVC Floating Action Button control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
<div id="target" style="min-height:200px; position:relative; width:300px; border:1px solid;">
 @Html.EJS().Fab("fab").Target("#target").Content("Add").Render()
</div>
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Floating Action Button control will be rendered in the default web browser.



### Event Click In Floating Action Button

The floating action button control triggers the `onclick` event when you click on the floating action button. You can use this event to perform the required action.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("e-icons e-edit").Content("Edit").Render()
<script>
 document.getElementById("fab").addEventListener('click', function () {
 alert("Edit is clicked!");
 });
</script>
```

#### ONCLICKEVENT.CS

```
public ActionResult OnClickEvent()
{
 return View();
}
```

### Icons in Asp.Net MVC Floating Action Button Control

You can customize the icon and text of Asp.Net MVC Floating Action Button(FAB) using [IconCss](#) and [Content](#) properties.

## FAB with icon

You can show icon only in Floating Action Button by setting [IconCss](#) property. You can show tooltip on hover to show additional details to end-user by setting [Title](#) attribute.

## CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("fab-icons fab-icon-people").Render()

<style>
 @@font-face {
 font-family: 'fab-icons';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAIAIAAwAgTlMvMj0kSSoAAAEoAAAVmNtYXCcV5yuAAABlAAAFRnbHlmHl6slgAAAFQAAASQaGVhZCG5vSMAAADQAAAAANmhoZWEHHowNkAAAArAAAACRobXR4E6AAAAAAAYAAAAUbG9jYQgKAywAAAHoAAAADG1heHABEGDDAAABCAAAACBuYW1l0KnKeQAABoQAAAI9cG9zdBh6gIAAAjEAAAARwABAAADUv9qAFoEAAAA//QD9AABAAAAAAAAAAAAAAAAABQABAAAAQAAZG1HNV8PPPUACwPoAAAAAN9TvcUAAAAA3108JQAAAAAD9AP0AAAACAACAAAAAAAAAAAAAFALCAAwAAAAAAGAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wTnDANS/2oAWgP0AJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEAEAAAAAKAAgAAgAC5wTnCoCK5wz//wA
A5wTnCoCK5wz//wAAAAAAAEACgAKAAoACgAAAAEAAgADAAQAAAAAABgA5AFyAkGAAQAAAAA
D6gPqAAsAAAEzESEVIREjESE1IQHDegGu/1J6/1IBrgPr/1J6/1IBrnoAAgAAAAADkwP0AHQatgA
AJRUjFSE1IzU/HjUjDxUvFSMVHx0DER8PPw8RLw8PDgHRIQF3ihISEhIRERAQDxAODg4NDQwLCws
JCQkHBwYGBAQDagJXAgIDBAQFBQYHBwgIEhUWFxoahB4eHx8eHhwaGhcWFRICACHBgUFBAQDagJ
XAgIDBAQGBgcICAKKCGsMDA0NDg8OEBAQEREREhMSdgECBQYICGoMDQ8PEBEREhMTEhEREhA8PDQw
KCgQHBQQAQIFBgJCgWNDhAQERETExMTEhEQDw8NDAsJBWYFARhbUVFbAgMDBAUFBgYHCAGICQo
KCgSLDAwMDQ0ODQ4PDg8PDxANDAsMCwWLCgSKCgKSEQ8NDAoHBgQBAQQAQGBwMDQ8REgkKCgSKCwW
LDAsMDRAPDw8ODw4NDg0NDAwMCwSKCgOJCAGIBWYGBQUEAwMCpP64EA8ODg0NCwsJCQCHBAQCAQE
CBAUGCAKJCwWMBw0ODg8BUBAPDg4NDQSLCQkHBgUEAgEBAgQFBgcJCQSLDQ0ODg8AAAAAAwAAAAA
DxgPoABAAIQBMAAABHgIUDgIiLgIOPgIyAR4CFA4CII4CND4CMicOAhUUFhcOAQcuASMiDgIVFbY
XDgMVMzQ+AjIeAhUzNC4CJz4BNTQ+AjIeAhUzNC4CJz4BNTQuAiIBYBgkFRUKMTcwJBuVJDA3Aak
YJBuVJDE3MCQVFSQwN2kkNiArJic9FBxWLy1JNiArJiI2JxVDIDZJUkk2IEMVJzYiJisgNklSSTY
gQxUnNiImKyA2SVICCwskMTcwJBuVJDA3MSQVAYYLJDE3MCQVFSQwNzEkFTMQNkkpL1YcFD0nJis
gNkkpL1YcETM+RyYpSTYgIDZJKSZHPjMRHFYvKUK2ICA2SSkmRz4zERxVMCLJNiAAAAADAAAAAP
0A/QAPwB/ALUAACUfDz8PLw8PDgUfDz8PLw8PDgMzEw8CFR8OITUhLwQ3IT8GEz8CNS8GIScJAsG
BAQIEBAUFBwcICAKJCgoKCgoKCQkICACHBQUEBAIBAQBAGQEBQUHBwgICQkKCgoKCgoJCQgIBWY
GBQQAeAgH+CwEBAGQEBQUHBwgICQkKCgoKCgoJCQgIBWcFBQQAeAgEBAQECBAQFBQcHCAgJCQoKCgo
KCgkJCAGHBgYFBAQCAclktUgIAQICBAQFBQcHCAgJCQkKCwJb/bsDAwIBASwBcQ8NDAwKCAi8AwQ
CAGMFBwgJCv0VK6ZwCgoKCQkICACGBgUEBAIBAQBAGQEBQUHBwgICQkKCgoKCgoJCQgIBWcFBQQ
EAgEBAQECBAQFBQcHCAgJCQoKCgoKCgkJCAGHBgYFBAQCAQEBAQIEBAUFBwcICAKJCgoKCgoKCQk
ICACHBQUEBAIBAQBAGQEBQUHBwgICQkKCgMW/on3JgWKCgoJCQgIBWYGBQQAeAgEBZAEBawIJVAE
CBQUHCQoBUAMHBRACKQgHBQMCZAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEACQABAAEAAAA
AAAIAABwAKAAEAAAAAAAAAMACQARAAEAAAAAAAAAQCAaAAEAAAAAAAAAUACwAjAAEAAAAAAAAAYACQAUAAE
AAAAAAAAoALAA3AAEAAAAAAAAASAEgBjAAMAAQQJAAAAAgB1AAMAAQQJAAEAEgB3AAMAAQQJAAIADgC
JAAMAAQQJAAAEgCXAAMAAQQJAAQAEgCpAAMAAQQJAAUAFgC7AAMAAQQJAAAYAEgDRAAMAAQQJAAo
AWADjAAMAAQQJAAAsAJAE7IEZhYi1JY29uc1JlZ3VsYXJGYWItSWNVbnNGYWIItSWNVbnNWZXJzaW9u
IDEuMEZhYi1JY29uc0ZvbG9zZ2VzZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAARgBhAGIALQBjJAGMabwBuAHMAUgBlAGcAdQBsAGEAcgBGAGEAYgAtAEkAYwBvAG4AcwBGAGEAYgAtAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAARgBhAGIALQBjJAGMabwBuAHMAARgBvAG4AdAAgAGcAZQBzAGkACAAAdQBzAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAA
AAAAABQECAQMBBAEFAQYAA2FkZANTaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('trueType');
 font-weight: normal;
 font-style: normal;
 }
```

```

[class^="fab-icon-"],
[class*=" fab-icon-"] {
 font-family: 'fab-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.fab-icon-people:before {
 content: "\e70a";
}
</style>

```

### FABICON.CS

```

public ActionResult FabIcon()
{
 return View();
}

```



### FAB with icon and text

You can show icon along with text in Floating Action Button by setting [IconCss](#) and [Content](#) properties.

### CSHTML

```

@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("fab-icons fab-icon-people").Content("Contacts").Render()
<style>
 @@font-face {
 font-family: 'fab-icons';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAKAIAAAwAgTlMvMj0kSSoAAAEoAAAVmNtYXCcV5yuAAABlAAAAFRnbHlmHl6slgAAAfQAAASQaGVhZCG5vSMAAADQAAAAANmhoZWEHovNkAAAArAAAACRobXR4E6AAAAAAAYAAAAUbg9jYQgKAYwAAAHoAAAADG1heHABEGDDAAABCAAAACBuYW1l0KnKeQAABOQAAAI9cG9zdBh6gIAAAjEAAAARwABAAADUv9qAFoEAAAA//QD9AABAAAAAAAAAAAAAAAAAAAAABQABAAAAQAAZG1HNV8PPPUACwPoAAAAAN9TvCUAAAAA31O8JQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAFALcAAwAAAAAAGAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wTnDANS/2oAWgP0AJYAAAABAAAAAAAAABAAAAAPoAAA

```



```

D6AAAA+gAAAPoAAAAAACAaaaaaAwAAABQAAwABAAAAFAAEAEAAAAAKAAgAAgAC5wTnCOCk5wz//wA
A5wTnCOCk5wz//wAAAAAAAAAAAAEACgAKAAoACgAAAAEAAgADAAQAAAAAABgA5AFyAkGAAQAAAAA
D6gPqAAsAAAEzESEVIREjESE1IQHDegGu/1J6/1IBrgPr/1J6/1IBrnoAAgAAAAADkwP0AHQAtgA
AJRUjFSE1IzU/HjUjDxUvFSMVHx0DER8PPw8RLw8PDgHRIQF3ihISEhIRERAQDxAODg4NDQwLCws
JCQkHBwYGBAQDagJXAgIDBAQFBQYHBwgIEhUWFxoaHB4eHx8eHhwaGhcWFR1ICAcHBgUFBAQDagJ
XAgIDBAQGBgcICAKKCGsMDA0NDg8OEBAQEREREhMSdgECBQYICgoMDQ8PEBEREhMTEhEREhA8PDQw
KCgQHBQQAQIFBgGJCgWNDhAQERETExMTEhEQDw8NDAsJBwYFARhbUVFbAgMDBAUFBgYHCAgICQo
KCgSLDAwMDQ0ODQ4PDg8PDxANDAsMCwWLCgSKCgkSEQ8NDAoHBgQBAQQGBwMDQ8REgkKCgSKCwW
LDAsMDRAPDw8ODw4NDg0NDAwMCwSKCgoJCAGIBWYGBQUEAwMCpP64EA8ODg0NCwsJCQCHBAQCAQE
CBAUGCAkJCwWMBw0ODg8BUBAPDg4NDQSLCQkHBgUEAgEBAGQFBgcJCQSLDQ0ODg8AAAAAAwAAAAA
DxgPoABAAIQBmAAABHgIUDgIiLgIOPgIyAR4CFA4CIi4CND4CMicOAhUUFhcOAQcuASMiDgIVFbY
XDgMVMzQ+AjIeAhUzNC4CJz4BNTQ+AjIeAhUzNC4CJz4BNTQuAiIBYBgkFRUKMTcWJBUVJDA3Aak
YJBUVJDE3MCQVFSQwN2kkNiArJic9FBxWLyLJNiArJiI2JxVDDIDZJUkk2IEMVJzYiJisgNklSSTY
gQxUnNiImKyA2SVICCwskMTcWJBUVJDA3MSQVAYYLJDE3MCQVFSQwNzEkFTMQNkkpL1YcFD0nJis
gNkkpL1YcETM+RyYpSTYgIDZJKSZHPjMRHFYvKUK2ICA2SSkmRz4zERxVMCLJNiAAAAADAAAAAP
0A/QAPwB/ALUAACUfDz8PLw8PDgUfDz8PLw8PDgMzEw8CFR8OITUhLwQ3IT8GEz8CNS8GIScjAsG
BAQIEBAUFBwcICAKJCgoKCgoKCQkICAcHBQUEBAIBAQBAGQEBQUHBwgICQkKCgoKCgoJCQgIBWY
GBQQEAgH+CwEBAGQEBQUHBwgICQkKCgoKCgoJCQgIBWcFBQQEAgEBAQECBAQFBQCHCAgJCQoKCgo
KCgkJCAGHBgYFBAQCAclktUgIAQICBAQFBQCHCAgJCQkKCwJb/bsDAwIBASwBcQ8NDAwKCAi8AwQ
CAGMFBwgJCv0VK6ZwCgoKCQkICAcGBgUEBAIBAQBAGQEBQUHBwgICQkKCgoKCgoJCQgIBWcFBQQ
EAgEBAQECBAQFBQCHCAgJCQoKCgoKCgkJCAGHBgYFBAQCAQEBAQIEBAUFBwcICAKJCgoKCgoKCQk
ICAcHBQUEBAIBAQBAGQEBQUHBwgICQkKCgMW/on3JgWKCgoJCQgIBWYGBQQEAgEBZAEBawIJVAE
CBQUHCQoBUAMHBRACKQgHBQMCZAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEACQABAAEAAAA
AAAIABwAKAAEAAAAAAAAAMACQARAAEAAAAAAAAAQCAaAAEAAAAAAAAAUACwAjAAEAAAAAAAAAYACQAUAAE
AAAAAAAAoALAA3AAEAAAAAAAAASAEgBjAAMAAQQJAAAAAgB1AAMAAQQJAAEAEgB3AAMAAQQJAAIADgC
JAAMAAQQJAAAEAgCXAAMAAQQJAAQAEgCpAAMAAQQJAAUAFgc7AAMAAQQJAAAYAEgDRAAMAAQQJAAo
AWADjAAMAAQQJAAASAJAE7IEZhYi1JY29uc1JlZ3VsYXJGYWItSWNvbnNGYWIItSWNvbnNWZXJzaW9
uIDEuMEZhYi1JY29uc0ZvbGQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGl
vd3d3LnN5bmNmdXNpb24uY29tACAARgBhAGIALQBJAGMABwBuAHMAUgBlAGcAdQBsAGEAcgBGAGE
AYgAtAEkAYwBvAG4AcwBGAGEAYgAtAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAARgB
hAGIALQBJAGMABwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAAdQBsAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB
5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAABQECAQMBBAEFAQYAA2FkZANtaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="fab-icon-"],
[class*=" fab-icon-"] {
 font-family: 'fab-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.fab-icon-people:before {
 content: "\e70a";
}
</style>

```

**FABTEXTICON.CS**

```
public ActionResult FabTextIcon()
{
 return View();
}
```


*Icon position*

You can change the position of icon when showing along with content by setting [IconPosition](#) property. By default, the icon is positioned on the left side together with text.

**CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("fab-icons fab-icon-people").Content("Contacts").IconPosition(IconPosition.Right).Render()
<style>
 @@font-face {
 font-family: 'fab-icons';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAKAIAAAwAgTlMvMj0kSSoAAAEoAAAAVmNtYXCcV5yuAAABlAAAAFRnbHlmHl6slgAAAFQAAASQaGVhZCG5vSMAAADQAAAAANmhoZWEHHowNkAAAArAAAACRobXR4E6AAAAAAYAAAAAUBg9jYQgKAYwAAAHoAAAADG1heHABEGDDAAABCAAAAACBuYW1l0KnKeQAABoQAAAI9cG9zdBh6gIAAAjEAAAARwABAAADUv9qAFoEAAAA//QD9AABAAAAAAAAAAAAAAAAAAAAABQABAAAAQAAZG1HNV8PPPUACwPoAAAAAN9TvcUAAAAA3108JQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAFALcAAwAAAAAAGAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wTnDANS/2oAWgP0AJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAEAAAAAKAAgAAgAC5wTnCOcK5wz//wA
A5wTnCOcK5wz//wAAAAAAAAAAAAEACgAKAAoACgAAAAEAAgADAAQAAAAABgA5AFyAkgAAQAAAAA
D6gPqAAsAAAEzESEVIREjESE1IQHDegGu/1J6/1IBrgPr/1J6/1IBrnoAAgAAAAADkwP0AHQAtgA
AJRUjFSE1IzU/HjUjDxUvFSMVHx0DER8PPw8RLw8PDgHRIQF3ihISEhIRERAQDxAODg4NDQwLCws
JCQkHBwYGBAQDagJXAgIDBAQFBQYHBwgIEhUWFxoahB4eHx8eHhwaGhcWFR1ICACHBgUFBAQDagJ
XAgIDBAQGBgcICAKKCGsMDA0NDg8OEBAQEREREhMSdgECBQYICgoMDQ8PEBEREhMTEhERE8PDQw
KCgQHBQQAQIFBggJCgWNDhAQERETExMTEhEQDw8NDAsJBwYFArhbUVFbAgMDBAUFBgYHCAGICQo
KCgsLDAwMDQ0ODQ4PDg8PDxANDAsMCwwLCgsKCgkSEQ8NDAoHBgQBAQQGBwMDQ8REgkKCgsKCww
LDAsMDRAPDw8ODw4NDg0NDawMCwsKCgoJCAGIBwYGBQUEAwMCpP64EA8ODg0NCwsJCQCHBAQCAQE
CBAUGCAkJCwwMBw0ODg8BUBAPDg4NDQsLCQkHBgUEAgEBAGQFBgcJCQsLDQ0ODg8AAAAAwAAAAA
DxgPoABAAIQBmAAABHgIUDgIiLgI0PgIyAR4CFA4CIi4CND4CMicOAhUUFhcOAQcuASMiDgIVFBY
XDgMVMzQ+AjIeAhUzNC4CJz4BNTQ+AjIeAhUzNC4CJz4BNTQuAiIBYBgkFRUkMTcwJBuVJDA3Aak
YJBuVJDE3MCQVFSQwN2kkNiArJic9FBxWLyLjNiArJiI2JxVDIDZJUkk2IEMVJzYiJisgNklSSTY
gQxUnNiImKyA2SVICcwskMTcwJBuVJDA3MSQVAYYLJDE3MCQVFSQwNzEkFTMQNkkpL1YcFD0nJis
gNkkpL1YcETM+RyYpSTYgIDZJKSZHPjMRHFYvKUK2ICA2SSkmRz4zERxVMclJNiAAAAADAAAAAP
0A/QAPwB/ALUAACUfDz8PLw8PDgUfDz8PLw8PDgMzEw8CFR8OITUhLwQ3IT8GEz8CNS8GIScJAsg
BAQIEBAUFBwcICAKJCgoKCgoKCQkICACHBQUEBAIBAQEBAgQEBQUHBwgICQkKCgoKCgoJCQgIBwY
```

```

GBQQEAgH+CwEBAGQEBQUHBWgICQkKCgokCgoJCQgIBwcFBQQEAgEBAQECBAQFBQcHCAgJCQoKCgo
KCgkJCAGHBgYFBAQCAclktUgIAQICBAQFBQcHCAgJCQkKCwJb/bsDAwIBASwBcQ8NDawKCAi8AwQ
CAGMFBWgJCv0VK6ZwCgoKCQkICAcGBgUEBAIBAQEBAgQEBQUHBWgICQkKCgokCgoJCQgIBwcFBQQ
EAgEBAQECBAQFBQcHCAgJCQoKCgoKCgkJCAGHBgYFBAQCAQEBAQIEBAUFBwcICakJCgoKCgoKCQk
ICAcHBQUEBAIBAQEBAgQEBQUHBWgICQkKCgMW/on3JgwKCgoJCQgIBwYGBQQEAgEBZAEBawIJVAE
CBQUHCQoBUAMHBRACKQgHBQMCZAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEACQABAAEAAAA
AAAIABwAKAAEAAAAAAAAAMACQARAAEAAAAAAAAAQCAaAAEAAAAAAAAUACwajAAEAAAAAAAAAYACQAAE
AAAAAAoALAA3AAEAAAAAAAAAEgBjAAMAAQQJAAAAAgB1AAMAAQQJAAEAEgB3AAMAAQQJAAIADgC
JAAMAAQQJAAAEgCXAAMAAQQJAAQAEgCpAAMAAQQJAAUAFgC7AAMAAQQJAAAYAEgDRAAMAAQQJAAo
AWADjAAMAAQQJAAASAJAE7IEZhYi1JY29uc1JlZ3VsYXJGYWItSWNvbnNGYWItSWNvbnNWZXJzaW9
uIDEuMEZhYi1JY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGl
vd3d3LnN5bmNmdXNpb24uY29tACAARgBhAGIALQBjAGMABwBuAHMAUgBlAGcAdQBsAGEAcgBGAGE
AYgAtAEkAYwBvAG4AcwBGAGEAYgAtAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAARgB
hAGIALQBjAGMABwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB
5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAA
AAAAABQECAQMBBAEFAQYAA2FkZANTaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="fab-icon-"],
[class*=" fab-icon-"] {
 font-family: 'fab-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.fab-icon-people:before {
 content: "\e70a";
}
</style>

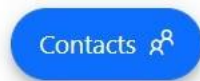
```

## ICONPOSITION.CS

```

public ActionResult IconPosition()
{
 return View();
}

```



## Styles in ASP.NET MVC Floating Action Button Control

This section explains the different styles of Floating Action Button.

### FAB styles

The ASP.NET MVC Floating Action Button supports the following predefined styles that can be defined using the [CssClass](#) property. You can customize by replacing the [CssClass](#) property with the below defined class.

Class	Description
-----	-----
e-success	Used to represent a positive action.
e-outline	Used to represent an appearance of button with outline.
e-info	Used to represent an informative action.
e-warning	Used to represent an action with caution.
e-danger	Used to represent a negative action.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("e-icons e-edit").CssClass("e-
warning").Render()
```

### FABSTYLES.CS

```
public ActionResult FabStyles()
{
 return View();
}
```



**Note:** Predefined Floating Action Button styles provide only the visual indication. So, Floating Action Button [Content](#) property should define the Floating Action Button style for the users of assistive technologies such as screen readers.

### Styles customization

To modify the Floating Action Button appearance, you need to override the default CSS of Floating Action Button component. Find the list of CSS classes and its corresponding section in Floating Action Button component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

|-----|-----|

|.e-fab.e-btn|To customize the FAB.|

|.e-fab.e-btn:hover|To customize the FAB on hover.|

|.e-fab.e-btn:focus|To customize the FAB on focus.|

|.e-fab.e-btn:active|To customize the FAB on active.|

|.e-fab.e-btn-icon|To customize the style of FAB icon.|

### Show text on hover

By using [CssClass](#), you can customize the Floating Action Button to show text on hover with applied transition effect.

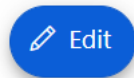
### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("e-icons e-edit").CssClass("fab-
hover").Content("Edit").Render()
<style>
 .e-fab.e-btn.fab-hover {
 padding: 6px 2px 8px 10px;
 }
 .fab-hover .text-container {
 overflow: hidden;
 width: 0;
 margin: 0;
 transition: width .5s linear 0s, margin .2s linear .5s;
```

```
}
.fab-hover:hover .text-container {
 width: 25px;
 margin-right:10px;
 transition: width .5s linear .2s, margin .2s linear 0s;
}
</style>
```

#### **FABHOVER.CS**

```
public ActionResult FabHover()
{
 return View();
}
```



#### Positions in Asp.Net MVC Floating Action Button Control

The floating action button can be positioned anywhere on the [Target](#) using the [Position](#) property. If the [Target](#) is not defined, then FAB is positioned based on the browser viewport.

The position values of Floating Action Button are as follows:

- TopLeft
- TopCenter
- TopRight
- MiddleLeft
- MiddleCenter
- MiddleRight
- BottomLeft
- BottomCenter
- BottomRight

Below example demonstrates Bottom Left position of FAB.

#### **CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("fab-icons fab-icon-
people").Position(FabPosition.BottomLeft).Render()
<style>
```

```

@@font-face {
 font-family: 'fab-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0kSSoAAAEoAAAAVmNtYXCcV5yuAAABlAAAAFRnbHlmHl6
slgAAAFQAAASQaGVhZCG5vSMAAADQAAAAANmhoZWEHowNkAAAArAAAACRobXR4E6AAAAAAYAAAA
UbG9jYQKGAYwAAAHoAAAADG1heHABEGDDAAABCAAAACBuYW1l0KnKeQAABoQAAAI9cG9zdBh6gIA
AAAJEAAAAARwABAAADUv9qAFoEAAAA//QD9AABAAAAAAAAAAAAAAAAAAAAABQABAAAAQAAZG1HNV8
PPPUACwPoAAAAAN9TvCUAAAAA3108JQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAFALCAAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wTnDANS/2oAWgP0AJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAFAAEAEAAAAAKAAgAAgAC5wTnCOCk5wz//wA
A5wTnCOCk5wz//wAAAAAAAAAAAAEACgAKAAoACgAAAAEAAgADAAQAAAAAABgA5AFyAkgaAAQAAAA
D6gPqAAAsAAAEzESeVIREjESE1IQHDEgGu/1J6/1IBrgPr/1J6/1IBrnoAAgAAAAADkwp0AHQAtgA
AJRUjFSE1IzU/HjUjDxUvFSMVHx0DER8PPw8RLw8PDgHRIQF3ihISEhIRERAQDxAODg4NDQwLCws
JCQkHBwYGBAQDagJXAgIDBAQFBQYHBwgIEhUWFxoaHB4eHx8eHhwaGhcWFR1ICaCHBgUFBAQDagJ
XAgIDBAQGBgcICakKCgsMDA0NDg8OEBAQEREREhMSdgECBQYICgoMDQ8PEBEREhMTEhEREhA8PDQw
KCgQHBQQAQIFBgJcGwNDhAQERETExMTEhEQDw8NDAsJBwYFARhbUVFbAgMDBAUFBgYHCAgICQo
KCgsLDAwMDQ0ODQ4PDg8PDxANDAsMCwWLCgsKCgkSEQ8NDAoHBgQBAQQGBwMDQ8REgkKCgsKCwW
LDAsMDRAPDw8ODw4NDg0NDAwMCwsKCgoJCAgIBwYGBQUEAwMCP64EA8ODg0NCwsJCQcHBAQCAQE
CBAUGCAkJCwWMBw0ODg8BUBAPDg4NDQsLCQkHBgUEAgEBAgQFBgcJCQsLDQ0ODg8AAAAAAwAAAA
DxgPoABAAIQBmAAABHgIUDgIiLgIOPgIyAR4CFA4CIi4CND4CMicOAhUUFhcOAQcuASMiDgIVFBY
XDgMVMzQ+AjIeAhUzNC4CJz4BNTQ+AjIeAhUzNC4CJz4BNTQuAiIBYBgkFRUKMTcwJBUVJDA3Aak
YJBUVJDE3MCQVFSQwN2kkNiArJic9FBxWLy1JNiArJiI2JxVDIDZJUkk2IEMVJzYiJisgNk1SSTY
gQxUnNiImKyA2SVICCwskMTcwJBUVJDA3MSQVAYYLJDE3MCQVFSQwNzEkFTMQNkkpL1YcFD0nJis
gNkkpL1YcETM+RyYpSTYgIDZJKSZHPjMRHFYvKUK2ICA2SSkmRz4zERxVMCL1JNiAAAAADAAAAAP
0A/QAPwB/ALUAACUfDz8PLw8PDgUfDz8PLw8PDgMzEw8CFR8OITUhLwQ3IT8GEz8CNS8GIScJAsG
BAQIEBAUFBwcICakJCgoKCgoKCQkICaCHBQUEBAIBAQBEBagQEBQUHBwgICQkKCgoKCgoJCQgIBwY
GBQQEAgH+CwEBAGQEBQUHBwgICQkKCgoKCgoJCQgIBwCQBQQEAgEBAQECEBAQFBQcHCAgJCQoKCgo
KCgkJCAgHBgYFBAQCAclktUgIAQICBAQFBQcHCAgJCQkKCwJb/bsDAwIBASwBcQ8NDAwKCAi8AwQ
CAGMFBwgJCv0VK6ZwCgoKCQkICAcGBgUEBAIBAQBEBagQEBQUHBwgICQkKCgoKCgoJCQgIBwCQBQQ
EAGEBAGQECBAQFBQcHCAgJCQoKCgoKCgkJCAgHBgYFBAQCAQEBAQIEBAUFBwcICakJCgoKCgoKCQk
ICAcHBQUEBAIBAQBEBAGQEBQUHBwgICQkKCgMW/on3JgwKCgoJCQgIBwYGBQQEAgEBZAEBawIJVAE
CBQUHCQoBUAMHBRACKQgHBQMCZAAAAAAAEgDeAAEAAAAAAAAAAAAQAAAAEAAAAAAAAEACQABAAEAAA
AAAIAABwAKAAEAAAAAAAAAMACQARAAEAAAAAAAAAQACQAAAEAAAAAAAAUACWajAAEAAAAAAAAAYACQAUAAE
AAAAAAAAoALAA3AAEAAAAAAAAASAEgBjAAMAAQQJAAAAAgB1AAMAAQQJAAEAEgB3AAMAAQQJAAIADgC
JAAMAAQQJAAMAEgCXAAMAAQQJAAQAEgCpAAMAAQQJAAUAFgC7AAMAAQQJAAAYAEgDRAAMAAQQJAAo
AWADjAAMAAQQJAAAsAJAE7IEZhYi1JY29uc1JlZ3VsYXJGYWItSWNvbmluZG91bnR1Z3V5b250cm8gU3R1ZG1
vd3d3LnN5bmNmdXNpb24uY29tACAARgBhAGIALQBJAGMABwBuAHMAUgBlAGcAdQBsAGEAcgBGAGE
AYgAtAEkAYwBvAG4AcwBGAGEAYgAtAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAARgB
hAGIALQBJAGMABwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAADQBzAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB
5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAA
AAAAABQECAQMBBAEFAQYAA2FkZANtaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="fab-icon-"],
[class*=" fab-icon-"] {
 font-family: 'fab-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;

```

```

 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 }
 .fab-icon-people:before {
 content: "\e70a";
 }
</style>

```

### **FABPOSITION.CS**

```

public ActionResult FabPosition()
{
 return View();
}

```



Below example demonstrates different supported positions of FAB.

### **CSHTML**

```

@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab1").Position(FabPosition.TopLeft).IconCss("fab-icons
fab-icon-people").Render()
@Html.EJS().Fab("fab2").Position(FabPosition.TopCenter).IconCss("fab-icons
fab-icon-people").Render()
@Html.EJS().Fab("fab3").Position(FabPosition.TopRight).IconCss("fab-icons
fab-icon-people").Render()
@Html.EJS().Fab("fab4").Position(FabPosition.MiddleLeft).IconCss("fab-icons
fab-icon-people").Render()
@Html.EJS().Fab("fab5").Position(FabPosition.MiddleCenter).IconCss("fab-
icons fab-icon-people").Render()
@Html.EJS().Fab("fab6").Position(FabPosition.MiddleRight).IconCss("fab-icons
fab-icon-people").Render()
@Html.EJS().Fab("fab7").Position(FabPosition.BottomLeft).IconCss("fab-icons
fab-icon-people").Render()
@Html.EJS().Fab("fab8").Position(FabPosition.BottomCenter).IconCss("fab-
icons fab-icon-people").Render()
@Html.EJS().Fab("fab9").Position(FabPosition.BottomRight).IconCss("fab-icons
fab-icon-people").Render()
<style>
 @@font-face {
 font-family: 'fab-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMj0kSSoAAAEoAAAAVmNtYXCcV5yuAAABlAAAAFRnbHlmHl6

```



```

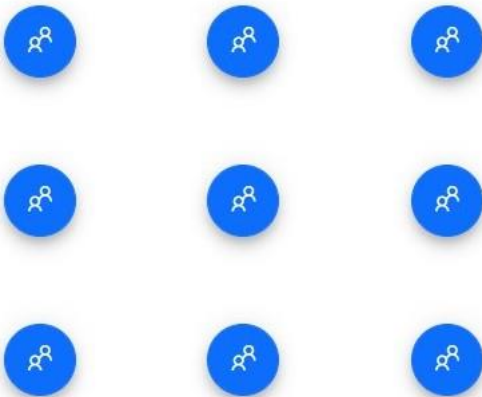
slgAAAFQAAASQaGVhZCG5vSMAAADQAAAAANmhoZWEHHowNkAAAArAAAACRobXR4E6AAAAAAYAAAAA
UbG9jYQKGAYwAAAHoAAAADG1heHABEGDDAAABCAAAACBuYW1l0KnKeQAABoQAAAI9cG9zdBh6gIA
AAAJEAAAARwABAAADUv9qAFoEAAAA//QD9AABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAZG1HNV8
PPPUACwPoAAAAAN9TvCUAAAAA3108JQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAFALCAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wTnDANS/2oAwgP0AJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAAACAAAAAwAAABQAAwABAAAFAAEEAEAAAAAKAAGAAgAC5wTnCOcK5wz//wA
A5wTnCOcK5wz//wAAAAAAAAAAAAAEACgAKAAoACgAAAAEEAgADAAQAAAAABG5A5FyAkGAAQAAAAA
D6gPqAAsAAAEzESEVIREjESE1IQHDegGu/1J6/1IBrgPr/1J6/1IBrnoAAgAAAAADkwP0AHQAtgA
AJRUjFSE1IzU/HjUjDxUvFSMVHx0DER8PPw8RLw8PDgHRIQF3ihISEhIRERAQDxAODg4NDQwLCws
JCQkHBWYGBAQDAGJXAgIDBAQFBQYHBWgIEhUWFxoahB4eHx8eHhwaGhcWFR1ICACHBgUFBAQDAGJ
XAgIDBAQGBgcICAKKCGsMDA0NDg8OEBAQEREREhMSdgECBQYICgoMDQ8PEBEREhMTEhEREhA8PDQw
KCgQBQCAQIFBgGJCgwNDhAQERETExMTEhEQDw8NDAsJBwYFARhbUVFbAgMDBAUFBgYHCAGICQo
KCgsLDAwMDQ0ODQ4PDg8PDxANDAsMCwwLCgsKCgkSEQ8NDAoHBgQBAQQGBwoMDQ8RegkKCgsKCww
LDAsMDRAPDw8ODw4NDg0NDawMCwsKCgoJCAGIBWYGBQUEAwMCP64EA8ODg0NCwsJCQCHBAQCAQE
CBAUGCAkJCwwMBw0ODg8BUBAPDg4NDQsLCQkHBgUEAgEBAGQFBgcJCQsLDQ0ODg8AAAAAAwAAAAA
DxgPoABAAIQBMAAABHGIUDgIiLgIOPgIyAR4CFA4CIi4CND4CMicOAhUUFhcOAQcuASMiDgIVFbY
XDgMVMzQ+AjIeAhUzNC4CJz4BNTQ+AjIeAhUzNC4CJz4BNTQuAiIBYBgkFRUKMTcwJBuVJDA3Aak
YJBuVJDE3MCQVFSQwN2kkNiArJic9FBxWLy1JNiArJiI2JxVDIDZJUkk2IEMVJzYiJisgNklSSTY
gQxUnNiImKyA2SVICcwsKMTcwJBuVJDA3MSQVAYYLJDE3MCQVFSQwNzEkFTMQNkkpL1YcFD0nJis
gNkkpL1YcETM+RyYpSTYgIDZJKSZHPjMRHFYvKUK2ICA2SSkmRz4zERxVMCLJNiAAAAADAAAAAP
0A/QAPwB/ALUAACUfDz8PLw8PDgUfDz8PLw8PDgMzEw8CFR8OITUhLwQ3IT8GEz8CNS8GIScJAsG
BAQIEBAUFBwcICAKJCgoKCgoKCQkICACHBQUEBAIBAQBEBAGQEBQUHBWgICQkKCgoKCgoJCQgIBwY
GBQQEAgH+CwEBAGQEBQUHBWgICQkKCgoKCgoJCQgIBwCFBQQEAgEBAQECAQFBQCHCAGJCQoKCgo
KCgkJCAGHBGyYFBAQCAclktUgIAQICBAQFBQCHCAGJCQkKCwJb/bsDAwIBASwBcQ8NDawKCAi8AwQ
CAGMFBWgJCv0VK6ZwCgoKCQkICACGBGUEBAIBAQBEBAGQEBQUHBWgICQkKCgoKCgoJCQgIBwCFBQQ
EAgEBAQECAQFBQCHCAGJCQoKCgoKCgkJCAGHBGyYFBAQCAQEBAQIEBAUFBwcICAKJCgoKCgoKCQk
ICACHBQUEBAIBAQBEBAGQEBQUHBWgICQkKCgMW/on3JgWKCgoJCQgIBWYGBQQEAgEBZAEBawIJVAE
CBQUHCQoBUAMHBRAKQgHBQMCZAAAAAAAEgDeAAEAAAAAAAAAAAAEAAAAAAAAEACQABAAEAAAAA
AAAIABwAKAAEAAAAAAAAAMACQARAAEAAAAAAAAAQACQAAAEAAAAAAAAAUACwAjAAEAAAAAAAAAYACQAAAE
AAAAAAAAoALAA3AAEAAAAAAAAAEGbJAAMAAQQAIAAAAGb1AAMAAQQAIAEAEAgB3AAMAAQQAIAAIADgC
JAAMAAQQAIAAMAEgCXAAMAAQQAIAAQAEgCpAAMAAQQAIAUAfGc7AAMAAQQAIAAYAEgDRAAMAAQQAIAAo
AWADJAAMAAQQAIAAsAJAE7IEZhYi1JY29uc1JlZ3VsYXJGYWItSWNVbnNGYWItSWNVbnNWZXJzaW9
uIDEuMEZhYi1JY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1
vd3d3LnN5bmNmdXNpb24uY29tACAARgBhAGIALQBJAGMABwBuAHMAUgB1AGcAdQBsaGEAcgBGAGE
AYgAtAEkAYwBvAG4AcwBGAGEAYgAtAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAARgB
hAGIALQBJAGMABwBuAHMARGBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAADQBzAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB
5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAA
AAAAABQECAQMBBAEFAQYAA2FkZANtaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="fab-icon-"],
[class*=" fab-icon-"] {
 font-family: 'fab-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}

```

```
.fab-icon-people:before {
 content: "\e70a";
}
</style>
```

### **FABPOSITION.CS**

```
public ActionResult FabPositions()
{
 return View();
}
```



### Custom position

You can define the custom position of the Floating Action Button by override the **top**, **left**, **right**, and **bottom** CSS properties using [CssClass](#).

### **CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("e-icons e-edit").CssClass("custom-
position").Render()
<style>
 .e-fab.e-btn.custom-position {
 left: 40px;
 top: 40px;
 bottom: unset;
 right: unset;
 }
</style>
```

### **CUSTOMPOSITION.CS**

```
public ActionResult CustomPosition()
{
 return View();
}
```



## Events in Floating Action Button Control

This section explains the available events in Floating Action Button Control.

### Created

Event triggers after the creation of Floating Action Button.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").Created("CreatedEvent()").IconCss("e-icons e-edit").Content("Edit").Render()
<script>
 function CreatedEvent() {
 // Your required action here
 }
</script>
```

#### CREATEDEVENT.CS

```
public ActionResult CreatedEvent()
{
 return View();
}
```

### OnClick

Event triggers when the Floating Action Button is clicked. Below example shows the Click event of the Floating Action Button.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().Fab("fab").IconCss("e-icons e-edit").Content("Edit").Render()
<script>
 document.getElementById("fab").addEventListener('click', function () {
 alert("Edit is clicked!");
 });
</script>
```

#### ONCLICKEVENT.CS

```
public ActionResult OnClickEvent()
{
 return View();
}
```

## Gantt

### Getting Started with ASP.NET MVC Gantt Control

This section briefly explains about how to include [ASP.NET MVC Gantt](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

<add namespace="Syncfusion.EJ2"/>

</namespaces>

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

**~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

**Register Syncfusion script manager**

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

**Add ASP.NET MVC Gantt Control**

Now, add the Syncfusion ASP.NET MVC Gantt control in `~/Views/Home/Index.cshtml` page.

Bind the data with Gantt control by using the [DataSource](#) property. It accepts an array of JavaScript object or the DataManager instance. The data source fields that are required to render the tasks are mapped to the Gantt control using the [TaskFields](#) property.

**CSHTML**

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)Model).Height("45
0px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
rogress("Progress").Child("SubTasks")).Render()
```

**HOMECONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(ganttData());
 }
}
```

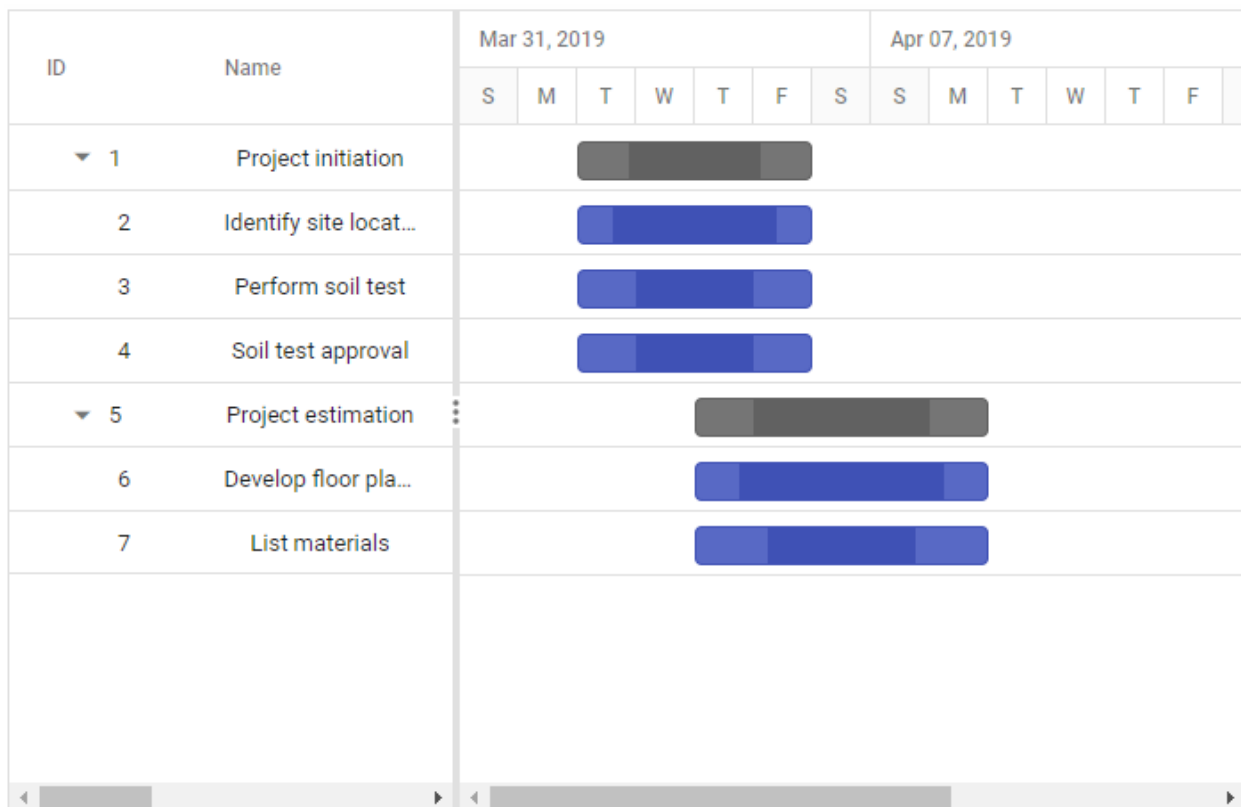
```
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
 List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
```

```

TaskId = 7,
TaskName = "List materials",
StartDate = new DateTime(2019, 04, 04),
Duration = 3,
Progress = 50
};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
}
public class GanttDataSource
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public DateTime StartDate { get; set; }
public DateTime EndDate { get; set; }
public int? Duration { get; set; }
public int Progress { get; set; }
public List<GanttDataSource> SubTasks { get; set; }
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Gantt control will be rendered in the default web browser.



### Defining columns

Gantt has an option to define columns as an array. You can customize the Gantt columns using the following properties:

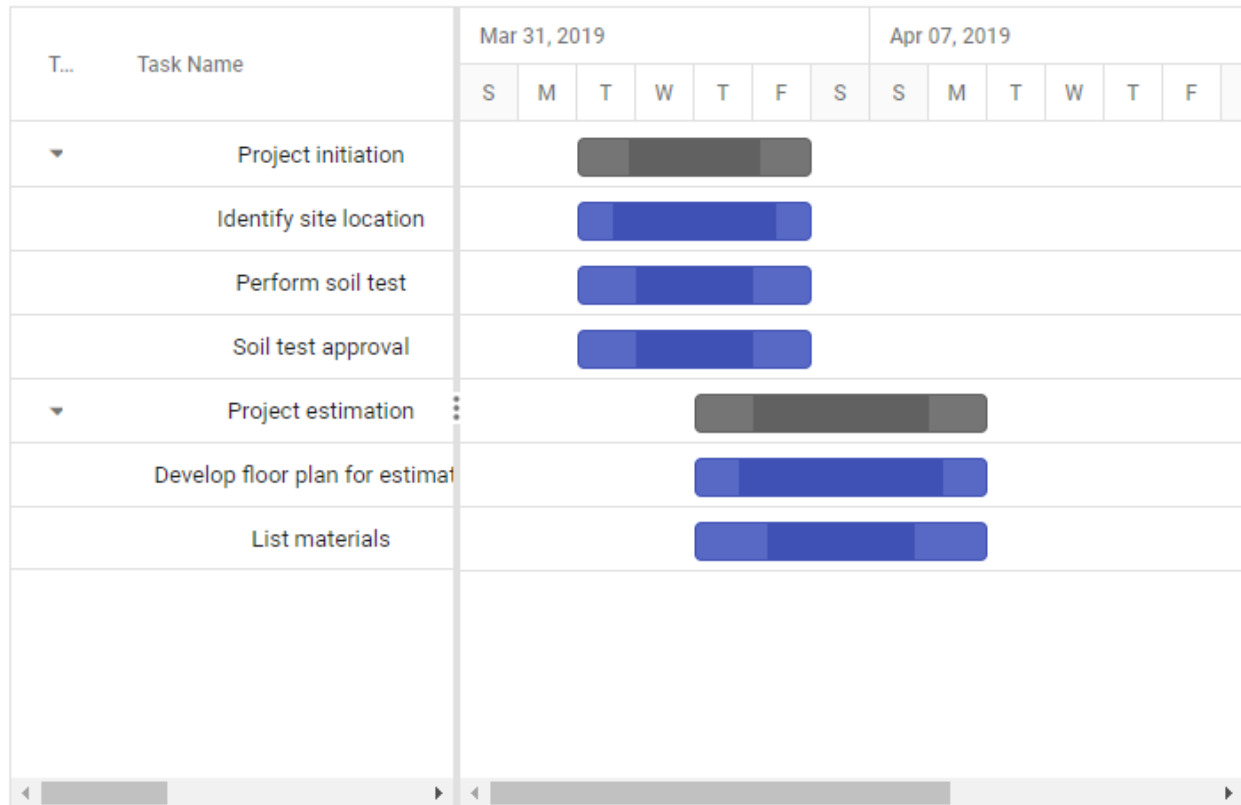
- [Field](#) : Maps the data source fields to the columns.
- [HeaderText](#) : Changes the title of columns.
- [TextAlign](#) : Changes the alignment of columns. By default, columns will be left aligned. To change the columns to right align, set [TextAlign](#) to right.
- [Format](#) : Formats the number and date values to standard or custom formats. Here, it is defined for the conversion of numeric values to currency.

### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)Model).Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(50).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(250).Add();
 col.Field("StartDate").HeaderText("Start Date").Add();
 col.Field("Duration").HeaderText("Duration").Add();
 col.Field("Progress").HeaderText("Progress").Format("C").Add();
}).Render()
```





### Enable editing

The editing feature enables you to edit the tasks in the Gantt control. It can be enabled by using the [EditSettings.AllowEditing](#) and [EditSettings.AllowTaskbarEditing](#) properties.

The following editing options are available to update the tasks in Gantt:

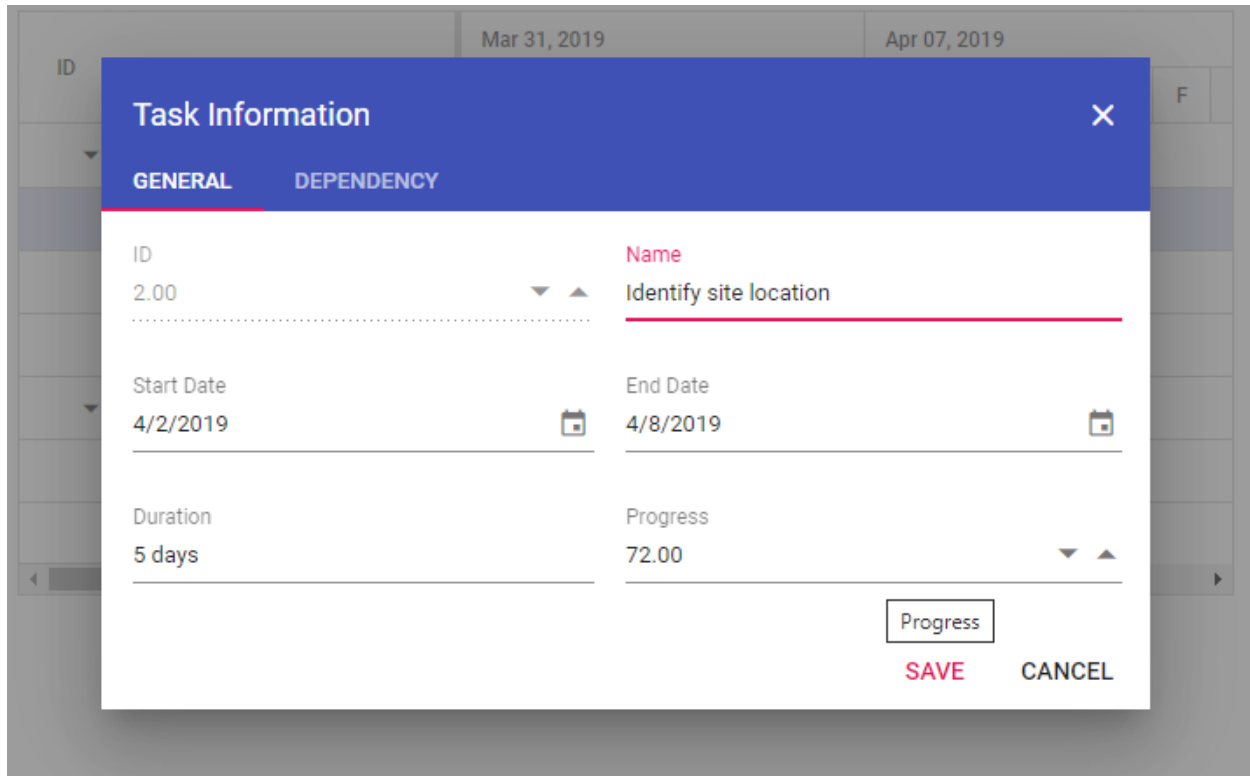
- Cell
- Dialog
- Taskbar
- Connector line

### Cell editing

Modify the task details through cell editing by setting the edit mode to **Auto**.

### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)Model).TaskFields(
 ts => ts.Id("TaskId").Name(
 "TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks")
).EditSettings(es =>
 es.AllowEditing(true).Mode(Syncfusion.EJ2.Gantt.EditMode.Auto)).Render()
```



**Note:** When the edit mode is set to **Auto**, you can change the cells to editable mode by double-clicking anywhere at the TreeGrid and edit the task details in the edit dialog by double-clicking anywhere at the chart.

#### Dialog editing

Modify the task details through dialog by setting the edit mode to **Dialog**.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)Model).TaskFields(
 ts => ts.Id("TaskId").Name("TaskName").StartDate(
 "StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency(
 "Predecessor").Child("SubTasks").EditSettings(es =>
 es.AllowEditing(true).Mode(Syncfusion.EJ2.Gantt.EditMode.Dialog)).Render()
```

**Note:** In dialog editing mode, the edit dialog will appear while performing double-click action in both TreeGrid and chart sides.

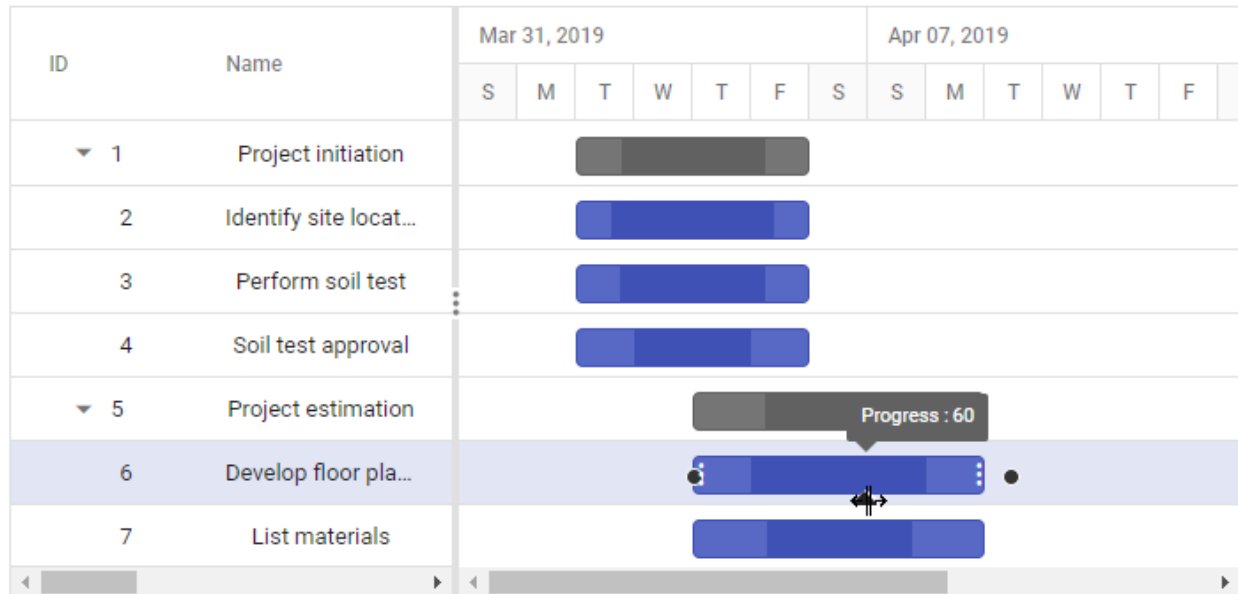
#### Taskbar editing

Modify the task details through user interaction such as resizing and dragging the taskbar by enabling the [AllowTaskbarEditing](#) property.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)Model).TaskFields(
 ts => ts.Id("TaskId").Name("TaskName").StartDate(
```

```
"StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks").EditSettings(es =>
 es.AllowTaskbarEditing(true)).Render()
```



#### Dependency editing

Modify the task dependencies using mouse interactions by enabling the [AllowTaskbarEditing](#) property along with mapping the task dependency data source field to the [Dependency](#) property.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)Model).TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate(
 "StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks").EditSettings(es =>
 es.AllowTaskbarEditing(true)).Render()
```

#### DEPENDENCYEDITING.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(ganttData());
 }
 public static List<GanttDataSource> ganttData()
 {
 List<GanttDataSource> GanttDataSourceCollection = new
 List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
```

```
StartDate = new DateTime(2019, 04, 02),
EndDate = new DateTime(2019, 04, 21),
SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
};
GanttDataSource Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Predecessor = "2FS"
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Predecessor = "3FS"
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Predecessor = "6SS"
```

```

};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string Predecessor { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

### Enable filtering

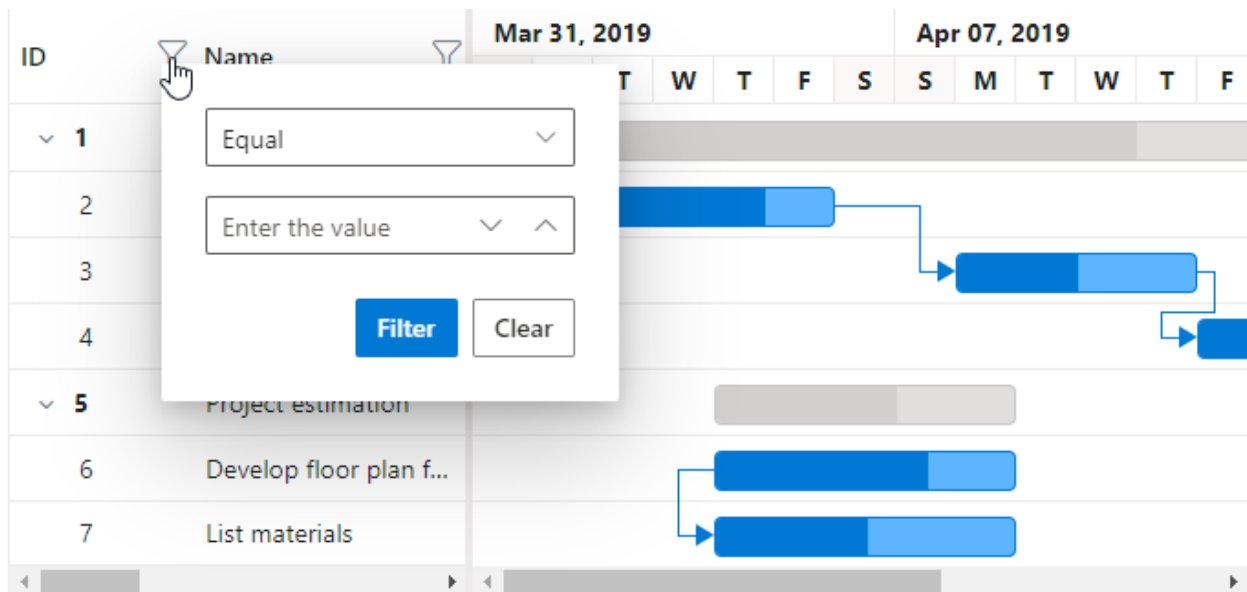
The filtering feature enables you to view the reduced amount of records based on filter criteria. Gantt provides the menu filtering support for each column. It can be enabled by setting the [AllowFiltering](#) property to **true**. Filtering feature can also be customized using the [FilterSettings](#) property.

### CSHTML

```

@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)Model).AllowFiltering(true).TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks")).Render()

```



### Enable sorting

The sorting feature enables you to order the records. It can be enabled by setting the [AllowSorting](#) property to `true`. The sorting feature can be customized using the [SortSettings](#) property.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)ViewBag.DataSource).AllowSorting(true).TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks")).Render()
```

### Sorting in ASP.NET MVC Gantt Control

#### Enabling predecessors or task relationships

Predecessor or task dependency in the Gantt control is used to depict the relationship between the tasks.

- Start to Start (SS): You cannot start a task until the dependent task starts.
- Start to Finish (SF): You cannot finish a task until the dependent task finishes.
- Finish to Start (FS): You cannot start a task until the dependent task completes.
- Finish to Finish (FF): You cannot finish a task until the dependent task completes.

You can show the relationship in tasks by using the [Dependency](#) property as shown in the following code example.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)Model).Height("450px").TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks")).Render()
```

### Assigning Resources

You can display and assign the resource for each task in the Gantt control. Create a collection of JSON object, which contains id, name, unit and group of the resources and assign it to the [Resources](#) property. Map these fields to the Gantt control using the [ResourceFields](#) property.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)Model).AllowFiltering(true).AllowSorting(true).Height("450px").TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks").ResourceInfo("ResourceId").LabelSettings(ls =>
 ls.RightLabel($"{if(ResourceId)} {ResourceId} {/if}"))
```

```
).ResourceFields(rf =>
rf.Id("ResourceId").Name("ResourceName")).Resources((IEnumerable<object>)ViewBag.projectResources).Render()
```

## HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.projectResources = projectResources();
 return View(ganttData());
 }
 public static List<GanttResources> projectResources()
 {
 List<GanttResources> GanttResourcesCollection = new List<GanttResources>();
 GanttResources Record1 = new GanttResources()
 {
 ResourceId = 1,
 ResourceName = "Martin Tamer"
 };
 GanttResources Record2 = new GanttResources()
 {
 ResourceId = 2,
 ResourceName = "Rose Fuller"
 };
 GanttResources Record3 = new GanttResources()
 {
 ResourceId = 3,
 ResourceName = "Margaret Buchanan"
 };
 GanttResources Record4 = new GanttResources()
 {
 ResourceId = 4,
 ResourceName = "Fuller King"
 };
 GanttResources Record5 = new GanttResources()
 {
 ResourceId = 5,
 ResourceName = "Davolio Fuller"
 };
 GanttResources Record6 = new GanttResources()
 {
 ResourceId = 6,
 ResourceName = "Van Jack"
 };
 GanttResources Record7 = new GanttResources()
 {
 ResourceId = 7,
 ResourceName = "Fuller Buchanan"
 };
 GanttResources Record8 = new GanttResources()
 {
 ResourceId = 8,
 ResourceName = "Jack Davolio"
 };
 }
}
```

```
GanttResources Record9 = new GanttResources()
{
 ResourceId = 9,
 ResourceName = "Tamer Vinet"
};
GanttResources Record10 = new GanttResources()
{
 ResourceId = 10,
 ResourceName = "Vinet Fuller"
};
GanttResources Record11 = new GanttResources()
{
 ResourceId = 11,
 ResourceName = "Bergs Anton"
};
GanttResources Record12 = new GanttResources()
{
 ResourceId = 12,
 ResourceName = "Construction Supervisor"
};
GanttResourcesCollection.Add(Record1);
GanttResourcesCollection.Add(Record2);
GanttResourcesCollection.Add(Record3);
GanttResourcesCollection.Add(Record4);
GanttResourcesCollection.Add(Record5);
GanttResourcesCollection.Add(Record6);
GanttResourcesCollection.Add(Record7);
GanttResourcesCollection.Add(Record8);
GanttResourcesCollection.Add(Record9);
GanttResourcesCollection.Add(Record10);
GanttResourcesCollection.Add(Record11);
GanttResourcesCollection.Add(Record12);
return GanttResourcesCollection;
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
 List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 1 },
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
```

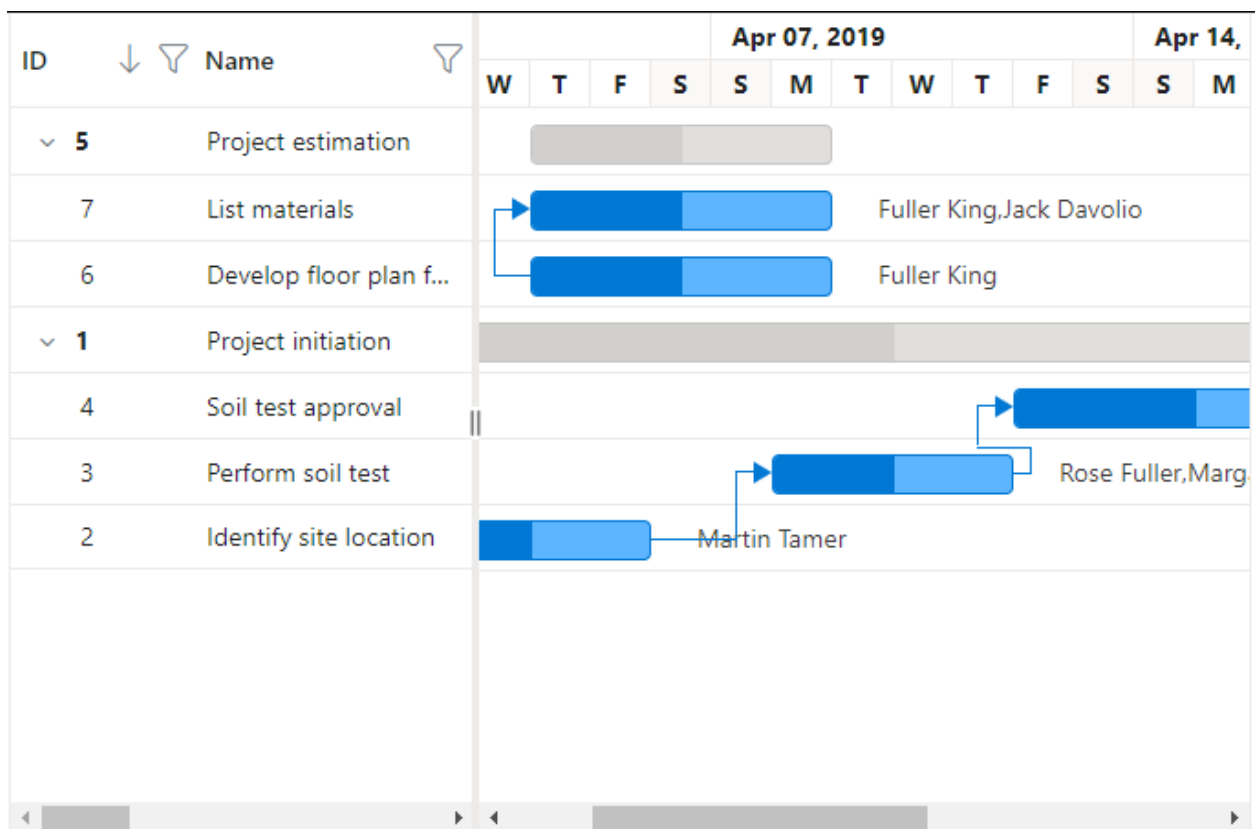


```
TaskId = 3,
TaskName = "Perform soil test",
StartDate = new DateTime(2019, 04, 02),
Duration = 4,
Progress = 50,
Predecessor = "2FS",
ResourceId = new int[] { 2, 3, 5 },
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Predecessor = "3FS"
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ResourceId = new int[] { 4 },
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Predecessor = "6SS",
 ResourceId = new int[] { 4, 8 },
};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
}
public class GanttResources
{
 public int ResourceId { get; set; }
}
```

```

public string ResourceName { get; set; }
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string Predecessor { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
}

```



**Note:** [View Sample in GitHub.](#)

### Data Binding in Gantt

The Gantt control uses **DataManager** for binding the data source, which supports both RESTful JSON data services and local JavaScript object array. The **DataSource** property can be assigned either with the instance of **DataManager** or JavaScript object array collection. The Gantt control supports binding two types of data:

- Local data
- Remote data

### Local data

To bind local data to Gantt, you can assign a JavaScript object array to the [DataSource](#) property. The local data source can also be provided as an instance of the [DataManager](#).

In local data binding, the data source for rendering the Gantt control is retrieved from the same application locally.

The following are the two types of data binding possible with the Gantt control:

- Hierarchical data binding.
- Self-referential data binding (Flat data).

### Hierarchical data binding

The [Child](#) property is used to map the child records in hierarchical data.

The following code example shows how to bind the hierarchical local data into the Gantt control.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id(
 "TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration(
 "Duration").Progress("Progress").Child("SubTasks")
).Render()
```

#### HIERARCHYDATA.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
 List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
```

```

 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
 }

 public class GanttDataSource
 {
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 }

```

```

 public List<GanttDataSource> SubTasks { get; set; }
 }

```

### Self-referential data binding (Flat data)

The Gantt control can be bound with self-referential data by mapping the data source field values to the [Id](#) and [ParentID](#) properties.

- **ID field:** This field contains unique values used to identify each individual task and it is mapped to the [Id](#) property.
- **Parent ID field:** This field contains values that indicate parent tasks and it is mapped to the [ParentID](#) property.

### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id(
 "TaskId").ParentId("ParentId").Name("TaskName").StartDate("StartDate").EndDa
te("EndDate").Duration("Duration").Progress("Progress")
).Render()

```

### SELFREFERENCE.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 ParentId = null
 });
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 ParentId = 1
 });
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 3,

```

```
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ParentId = 1
 });
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ParentId = 1
 });
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 ParentId = null
 });
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70,
 ParentId = 5
 });
 GanttDataSourceCollection.Add(new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ParentId = 5
 });
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public int? ParentId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
}
```

### Remote data

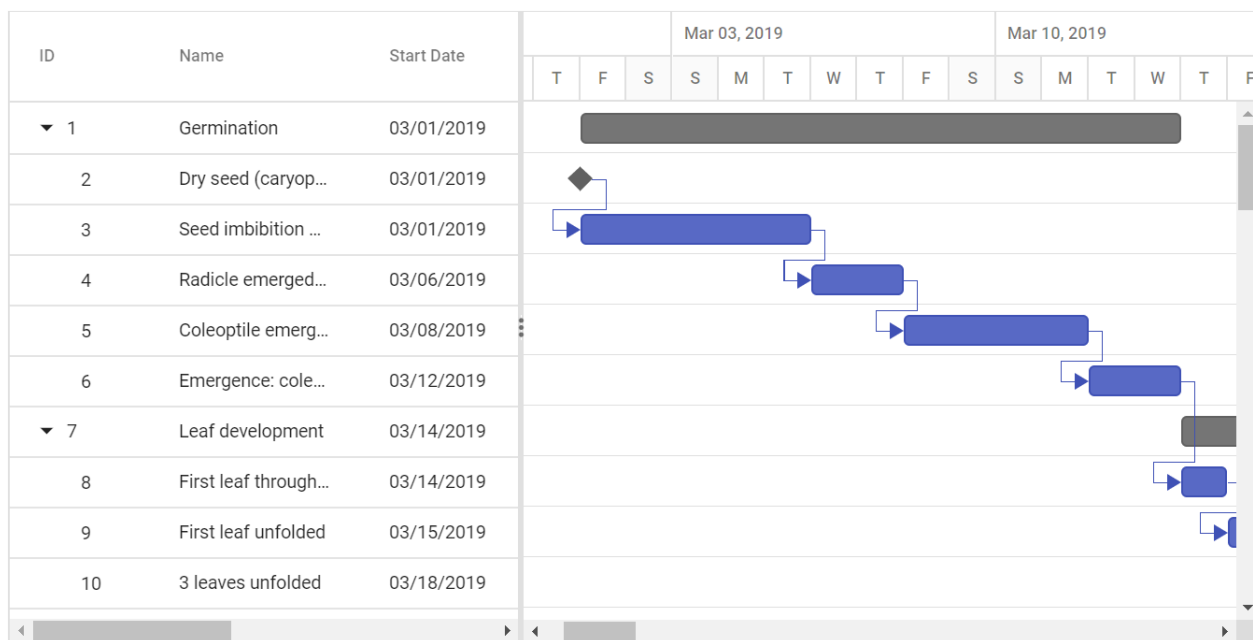
To bind remote data to the Gantt component, assign service data as an instance of **DataManager** to the [DataSource](#) property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource(dataManger =>
{
 dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/GanttData").CrossDomain(true).Adaptor("WebApiAdaptor");
 }).Height("450px").TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").Progress("Progress").Duration("Duration")
).Dependency("Predecessor").Child("SubTasks").Render()
```

### REMOTEDATA.CS

```
public IActionResult Index()
{
 return View();
}
```



### URL Adaptor

In Gantt, we can fetch data from SQL database using **ADO.NET** Entity Data Model and update the changes on CRUD action to the server by using **DataManager** support. To communicate with the remote data we are using **UrlAdaptor** of **DataManager** property to call the server method and get back resultant data in JSON format. We can know more about **UrlAdaptor** from [here](#).

**Note:** Refer the [link](#) to create the **ADO.NET** Entity Data Model in Visual Studio,

We can define data source for Gantt as instance of DataManager using **url** property of DataManager. Check the below code snippet to assign data source to Gantt.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource(dataManager =>
{
 dataManager.Url("/Home/UrlDatasource").Adaptor("UrlAdaptor")
}).TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Render()
```

### URLADAPTOR.CS

```
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult UrlDatasource(DataManagerRequest dm)
{

 List<GanttData>DataList = db.GanttDatas.ToList();
 var count = DataList.Count();
 return Json(new { result = DataList, count = count });
}
```

### *Remote Save Adaptor*

You may need to perform all Gantt Actions on the client-side except the CRUD operations, which should be interacted with the server-side to persist data. It can be achieved in Gantt by using

### **RemoteSaveAdaptor.**

Datasource must be set to the **json** property and set **RemoteSaveAdaptor** to the **adaptor** property of DataManager. CRUD operations can be mapped to the server-side using the **batchUrl** properties.

You can use the following code example to use **RemoteSaveAdaptor** in Gantt.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource(dataManager =>
{
 dataManager.Url("/Home/BatchUpdate").Adaptor("remoteSaveAdaptor")
}).TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Render()
```

### REMOTESAVEADAPTOR.CS

```
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchUpdate(DataManagerRequest dm)
{

 List<GanttData>DataList = db.GanttDatas.ToList();
 var count = DataList.Count();
 return Json(new { result = DataList, count = count });
}
```



```
}

```

The following code example describes the CRUD operations handled at server-side.

```
`json
public IActionResult BatchUpdate([FromBody] CRUDModel batchmodel)
{
 public class CRUDModel
 {
 public List<GanttDataSource> added { get; set; }
 public List<GanttDataSource> changed { get; set; }
 public List<GanttDataSource> deleted { get; set; }
 public object key { get; set; }
 public string action { get; set; }
 public string table { get; set; }
 }

 public IActionResult BatchUpdate([FromBody] CRUDModel batchmodel)
 {
 if (batchmodel.changed != null)
 {
 for (var i = 0; i < batchmodel.changed.Count(); i++)
 {
 var value = batchmodel.changed[i];
 GanttDataSource result = DataList.Where(or => or.taskId == value.taskId).FirstOrDefault();
 result.taskId = value.taskId;
 result.taskName = value.taskName;
 result.startDate = value.startDate;
 result.endDate = value.endDate;
 result.duration = value.duration;
 result.progress = value.progress;
 result.parentID = value.parentID;
 }
 }
 if (batchmodel.deleted != null)
 {

```

```

for (var i = 0; i < batchmodel.deleted.Count(); i++)
{
 DataList.Remove(DataList.Where(or => or.taskId.Equals(batchmodel.deleted[i].taskId)).FirstOrDefault());
 RemoveChildRecords(batchmodel.deleted[i].taskId);
}
}

if (batchmodel.added != null)
{
 for (var i = 0; i < batchmodel.added.Count(); i++)
 {
 DataList.Add(batchmodel.added[i]);
 }
}

return Json(new { addedRecords = batchmodel.added, changedRecords = batchmodel.changed,
 deletedRecords = batchmodel.deleted });
}

public void RemoveChildRecords(int key)
{
 var childList = DataList.Where(x => x.parentID == key).ToList();
 foreach (var item in childList)
 {
 DataList.Remove(item);
 RemoveChildRecords(item.taskId);
 }
}

return Json(new { addedRecords = batchmodel.added, changedRecords = batchmodel.changed,
 deletedRecords = batchmodel.deleted });
}

```

#### *Load child on demand*

To render child records on demand, assign a remote service URL in the instance of DataManager to the Url property. To interact with the remote data source, provide the endpoint URL and also define the [HasChildMapping](#) property in taskFields of Gantt Chart.

The `hasChildMapping` property maps the field name in the data source, which denotes whether the current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

When [LoadChildOnDemand](#) is disabled, all the root nodes are rendered in a collapsed state at initial load. On expanding the root node, the child nodes will be loaded from the remote server.

When `EnableVirtualization` is enabled and `LoadChildOnDemand` is disabled, only the current viewport root nodes are rendered in a collapsed state.

When a root node is expanded, its child nodes are rendered and maintained in a collection locally, such that on consecutive expand/collapse actions on the root node, the child nodes are loaded locally instead of from the remote server.

When the `LoadChildOnDemand` is enabled, parent records are rendered in an expanded state.

### **CSHTML**

```
@Html.EJS().Gantt("DefaultFunctionalities").DataSource(dataManger =>
{
 dataManger.Url("https://services.syncfusion.com/aspnet/development/api/GanttLoadOnDemand").CrossDomain(true).Adaptor("WebApiAdaptor");
})

.Height("460px").EnableVirtualization(true).LoadChildOnDemand(false)
.TaskFields(ts =>
ts.Id("taskId").Name("taskName").StartDate("startDate").Progress("progress")

.Duration("duration").ParentID("parentID").EndDate("endDate").HasChildMapping("isParent"))

.Columns(col =>
{
 col.Field("taskId").Add();
 col.Field("taskName").Add();
 col.Field("startDate").Add();
 col.Field("duration").Add();
})
.TreeColumnIndex(1)
.AllowSelection(true)
.HighlightWeekends(true)
.IncludeWeekend(true)
.ProjectStartDate("01/02/2000")
.ProjectEndDate("12/01/2002")
.Render()
```

### **LOADCHILDONDEMAND.CS**

```
public IActionResult Index()
{
 return View();
}
```

The following code example describes handling of Load on demand at server end.

```
`json
public object Get()
{
 DataOperations operation = new DataOperations();
 var queryString = Request.Query;
 if (tree.Count == 0)
 tree = TreeData.GetTree();
 if (queryString.Keys.Contains("$filter") && !queryString.Keys.Contains("$stop"))
 {
 StringValues filter;
 queryString.TryGetValue("$filter", out filter);
 int? fltr;
 if (filter[0].ToString().Split("eq")[1] == " null")
 {
 fltr = null;
 }
 else
 {
 fltr = Int32.Parse(filter[0].ToString().Split("eq")[1]);
 }
 IQueryable<TreeData> data1 = tree.Where(f => f.parentID == fltr).AsQueryable();
 return new { result = data1.ToList(), count = data1.Count() };
 }
 StringValues expand;
 queryString.TryGetValue("$expand", out expand);
 if (queryString.Keys.Contains("$expand")) // setting the ExpandStateMapping property whether is true
 or false
 {
 if (expand[0].ToString().Split(",")[0] == "ExpandingAction")
 {
 var val = TreeData.GetTree().Where(ds => ds.taskId ==
 int.Parse(expand[0].ToString().Split(",")[1])).FirstOrDefault();
 val.IsExpanded = true;
 }
 }
}
```

```
}
else if (expand[0].ToString().Split(",")[0] == "CollapsingAction")
{
 var val = TreeData.GetTree().Where(ds => ds.taskId ==
int.Parse(expand[0].ToString().Split(",")[1])).FirstOrDefault();
 val.IsExpanded = false;
}
}

List<TreeData> data = tree.ToList();
if (queryString.Keys.Contains("$select"))
{
 data = (from ord in tree
select new TreeData
{
 parentID = ord.parentID
}
).ToList();
 return data;
}
data = data.Where(p => p.parentID == null).ToList();
int count = data.Count;
if (queryString.Keys.Contains("$inlinecount"))
{
 StringValues Skip;
 StringValues Take;
 StringValues loadchild;
 int skip = (queryString.TryGetValue("$skip", out Skip)) ? Convert.ToInt32(Skip[0]) : 0;
 int top = (queryString.TryGetValue("$top", out Take)) ? Convert.ToInt32(Take[0]) : data.Count();
 var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.parentID)
 .Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
 foreach (var Record in data.ToList())
 {
 if (GroupData.ContainsKey(Record.taskId.ToString()))
```

```
{
var ChildGroup = GroupData[Record.taskId.ToString()];
if (ChildGroup?.Count > 0)
AppendChildren(ChildGroup, Record, GroupData, data);
}
}

if (expand.Count > 0 && expand[0].ToString().Split(",")[0] == "CollapsingAction")
{
string IdMapping = "taskId";
List<WhereFilter> CollapseFilter = new List<WhereFilter>();
CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = expand[0].ToString().Split(",")[1],
Operator = "equal" });
var CollapsedParentRecord = operation.PerformFiltering(data, CollapseFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
skip = index;
}

else if (expand.Count > 0 && expand[0].ToString().Split(",")[0] == "ExpandingAction")
{
string IdMapping = "taskId";
List<WhereFilter> ExpandFilter = new List<WhereFilter>();
ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = expand[0].ToString().Split(",")[1],
Operator = "equal" });
var ExpandedParentRecord = operation.PerformFiltering(data, ExpandFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
skip = index;
}

return new { result = data.Skip(skip).Take(top), count = data.Count };
}

else
{
return TreeData.GetTree();
}
}
```

```
private void AppendChildren(List<TreeData> ChildRecords, TreeData ParentItem, Dictionary<string,
List<TreeData>> GroupData, List<TreeData> data)
{
 var queryString = Request.Query;
 string TaskId = ParentItem.taskId.ToString();
 var index = data.IndexOf(ParentItem);
 foreach (var Child in ChildRecords)
 {
 string ParentId = Child.parentID.ToString();
 if (TaskId == ParentId && (bool)ParentItem.IsExpanded)
 {
 if (data.IndexOf(Child) == -1)
 {
 ((IList)data).Insert(++index, Child);
 }
 if (GroupData.ContainsKey(Child.taskId.ToString()))
 {
 var DeepChildRecords = GroupData[Child.taskId.ToString()];
 if (DeepChildRecords?.Count > 0)
 {
 AppendChildren(DeepChildRecords, Child, GroupData, data);
 }
 }
 }
 }
}

// GET: api/Orders/
[HttpGet("{id}", Name = "Get")]
public string Get(int id)
{
 return "value";
}

[HttpPost]
public object Post([FromBody] TreeData[] value)
{
 //handle insert action
 for (var i = 0; i < value.Length; i++)
```

```
{
tree.Insert(0, value[i]);
}
return value;
}
//// PUT: api/Orders
[HttpPut]
public object Put([FromBody] TreeData[] value)
{
//handle edit action
if (value.Length == 1 && value[0].isParent == true)
{
UpdateDependentRecords(value[0]);
}
for (var i = 0; i < value.Length; i++) {
var ord = value[i];
TreeData val = tree.Where(or => or.taskId == ord.taskId).FirstOrDefault();
val.taskId = ord.taskId;
val.taskName = ord.taskName;
val.endDate = ord.endDate;
val.startDate = ord.startDate;
val.duration = ord.duration;
val.predecessor = ord.predecessor;
}
return value;
}
private void UpdateDependentRecords(TreeData ParentItem)
{
var data = tree.Where(p => p.parentID == ParentItem.taskId).ToList();
var previousData = tree.Where(p => p.taskId == ParentItem.taskId).ToList();
var previousStartDate = previousData[0].startDate;
var previousEndDate = previousData[0].endDate;
```



```
double sdiff = (double)GetTimeDifference((DateTime)previousStartDate,
(DateTime)ParentItem.startDate);

double ediff = (double)GetTimeDifference((DateTime)previousEndDate,
(DateTime)ParentItem.endDate);

GetRootChildRecords(ParentItem);
for(var i=0; i<ChildRecords.Count;i++)
{
 ChildRecords[i].startDate = ((DateTime)ChildRecords[i].startDate).AddSeconds(sdiff);
 ChildRecords[i].endDate = ((DateTime)ChildRecords[i].endDate).AddSeconds(ediff);
}
}

private void GetRootChildRecords(TreeData ParentItem)
{
 var currentchildRecords = tree.Where(p => p.parentID == ParentItem.taskId).ToList();
 for (var i = 0; i < currentchildRecords.Count; i++) {
 var currentRecord = currentchildRecords[i];
 ChildRecords.Add(currentRecord);
 if (currentRecord.isParent == true)
 {
 GetRootChildRecords(currentRecord);
 }
 }
}

public object GetTimeDifference(DateTime sdate, DateTime edate)
{
 return new DateTime(edate.Year, edate.Month, edate.Day, edate.Hour, edate.Minute, edate.Second,
DateTimeKind.Utc).Subtract(new DateTime(sdate.Year, sdate.Month, sdate.Day, sdate.Hour,
sdate.Minute, sdate.Second, DateTimeKind.Utc)).TotalSeconds;
}

// DELETE: api/ApiWithActions
[HttpDelete("{id:int}")]
[Route("Orders/{id:int}")]
public object Delete(int id)
{
}
```

```
//handle delete action
tree.Remove(tree.Where(or => or.taskId == id).FirstOrDefault());
return Json(id);
}

public class CRUDModel<T> where T : class
{
 public TreeData Value;
 public int Key { get; set; }
 public int RelationalKey { get; set; }
 public List<TreeData> added { get; set; }
 public List<TreeData> changed { get; set; }
 public List<TreeData> deleted { get; set; }
}

public class TreeData
{
 public static List<TreeData> tree = new List<TreeData>();
 [System.ComponentModel.DataAnnotations.Key]
 public int taskId { get; set; }
 public string taskName { get; set; }
 public DateTime startDate { get; set; }
 public DateTime endDate { get; set; }
 public string duration { get; set; }
 public int progress { get; set; }
 public int? parentID { get; set; }
 public string predecessor { get; set; }
 public bool? isParent { get; set; }
 public bool? IsExpanded { get; set; }
 public static List<TreeData> GetTree()
 {
 if (tree.Count == 0)
 {
 Random rand = new Random();
 var x = 0;
```

```
int duration = 0;
DateTime startDate = new DateTime(2000, 1, 3, 08, 00, 00);
for (var i = 1; i <= 50; i++)
{
 startDate = startDate.AddDays(i == 1 ? 0 : 7);
 DateTime childStartDate = startDate;
 TreeData Parent = new TreeData()
 {
 taskId = ++x,
 taskName = "Task " + x,
 startDate = startDate,
 endDate = childStartDate.AddDays(26),
 duration = "20",
 progress = rand.Next(100),
 predecessor = null,
 isParent = true,
 parentID = null,
 IsExpanded = false
 };
 tree.Add(Parent);
 for (var j = 1; j <= 4; j++)
 {
 childStartDate = childStartDate.AddDays(j == 1 ? 0 : duration + 2);
 duration = 5;
 tree.Add(new TreeData()
 {
 taskId = ++x,
 taskName = "Task " + x,
 startDate = childStartDate,
 endDate = childStartDate.AddDays(5),
 duration = duration.ToString(),
 progress = rand.Next(100),
 parentID = Parent.taskId,
```

```

predecessor = (j > 1 ? (x - 1) + "FS" : ""),
isParent = false,
IsExpanded = false
});
}
}
}
return tree;
}
}
,

```

### Limitations

- Filtering, sorting and searching are not supported in load on demand.
- Only Self-Referential type data is supported with remote data binding in Gantt Chart.
- Load-on-demand supports only the validated data source

### Sending additional parameters to the server

We can pass additional parameters using [addParams](#) method of [Query](#) class. In server side we have inherited and shown the additional parameter value in Syncfusion DataManager class itself. We pass an additional parameter in load time using [load](#) event. We can also pass additional parameter to the CRUD model. Check the below code snippet to send additional parameter to Gantt.

### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource(dataManager =>
{
 dataManager.Url("http://localhost:50039/Home/UrlDatasource").Adaptor("UrlAdaptor").BatchUrl("http://localhost:50039/Home/BatchSave");
}).Toolbar(new List<string>()
{ "Add", "Cancel", "CollapseAll", "Delete", "Edit", "ExpandAll", "Update"
}).TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").ParentId("ParentId")

).EditSettings(es=>es.AllowEditing(true).AllowAdding(true).AllowDeleting(true)).Load("load").Render()
<script>
function load(args) {
 var ganttObj = document.getElementById("Gantt").ej2_instances[0];
 ganttObj.query = new Query().addParams('ej2Gantt', "test");
}
</script>

```

### SERVERPARAMETERS.CS

```

namespace URLAdaptor.Controllers
{
 public class HomeController : Controller
 {
 ...///
 //inherit the class to show age as property of DataManager
 public class Test : DataManagerRequest
 {
 public string ej2Gantt { get; set; }
 }
 public ActionResult UrlDatasource([FromBody]Test dm)
 {
 if (DataList == null)
 {
 ProjectData datasource = new ProjectData();
 DataList = datasource.GetUrlDataSource();
 }
 var count = DataList.Count();
 return Json(new { result = DataList, count = count },
 JsonRequestBehavior.AllowGet);
 }
 ...///
 public class ICRUDModel<T> where T : class
 {
 public object key { get; set; }
 public T value { get; set; }
 public List<T> added { get; set; }
 public List<T> changed { get; set; }
 public List<T> deleted { get; set; }
 public IDictionary<string, object> @params { get; set; }
 }
 ...///
 }
}

```

### Handling HTTP error

During server interaction from the Gantt, some server-side exceptions may occur, and you can acquire those error messages or exception details in client-side using the [actionFailure](#) event.

The argument passed to the `actionFailure` event contains the error details returned from the server.

### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource(dataManager =>
{
 dataManager.Url("http://some.com/invalidUrl")).TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").ParentId("ParentId")
).ActionFailure("actionFailure").Render()
<script>
function actionFailure(args) {
 var ganttObj = document.getElementById("Gantt").ej2_instances[0];
 let span: HTMLElement = document.createElement('span');
 ganttObj.element.parentNode.insertBefore(span, ganttObj.element);
 span.style.color = '#FF0000'
}

```

```
span.innerHTML = 'Server exception: 404 Not found';
</script>
```

### HTTPERROR.CS

```
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult UrlDatasource(DataManagerRequest dm)
{
 List<GanttData>DataList = db.GanttDatas.ToList();
 var count = DataList.Count();
 return Json(new { result = DataList, count = count });
}
```

### Binding with Fetch

You can use Gantt [dataSource](#) property to bind the data source to Gantt from external Fetch request. In the below code we have fetched the data source from the server with the help of Fetch request and provided that to [dataSource](#) property by using [onSuccess](#) event of the Fetch.

### CSHTML

```
@Html.EJS().Button("dataBind").Content("Bind Data").CssClass("e-
primary").Render()
@Html.EJS().Gantt("Gantt").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").Child("SubTasks")
).Render()
<script>
 document.getElementById('dataBind').addEventListener('click',
function (args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 var fetch = new
Fetch("https://ej2services.syncfusion.com/production/web-
services/api/GanttData", "GET");
 ganttObj.showSpinner();
 fetch.send();
 fetch.onSuccess = function (data: any) {
 ganttObj.hideSpinner();
 ganttObj.dataSource = data.Items;
 ganttObj.refresh();

 };
 });
</script>
```

### AJAXBINDING.CS

```
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult UrlDatasource(DataManagerRequest dm)
{
 List<GanttData>DataList = db.GanttDatas.ToList();
 var count = DataList.Count();
 return Json(new { result = DataList, count = count });
}
```

```
}
```

**Note:** If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

### Split task

The **Split-task** feature allows you to split a task or interrupt the work during planned or unforeseen circumstances. We can split the task either in load time or dynamically, by defining the segments either in hierarchical or self-referential way.

### Hierarchical

To split a task at load time in hierarchical way, we need to define the segment details in datasource and this field should be mapped by using the [taskFields.Segments](#) property.

```
`html
```

```
GanttDataSource Record2Child1 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2019, 02, 04),
 EndDate = new DateTime(2019, 02, 10),
 Duration = 10,
 Progress = 60,
 Segments = new List<GanttSegment>
 {
 new GanttSegment {StartDate = new DateTime(2019,02,04), Duration = 2},
 new GanttSegment {StartDate = new DateTime(2019,02,05), Duration = 5},
 new GanttSegment {StartDate = new DateTime(2019,02,08), Duration = 3}
 }
};
```

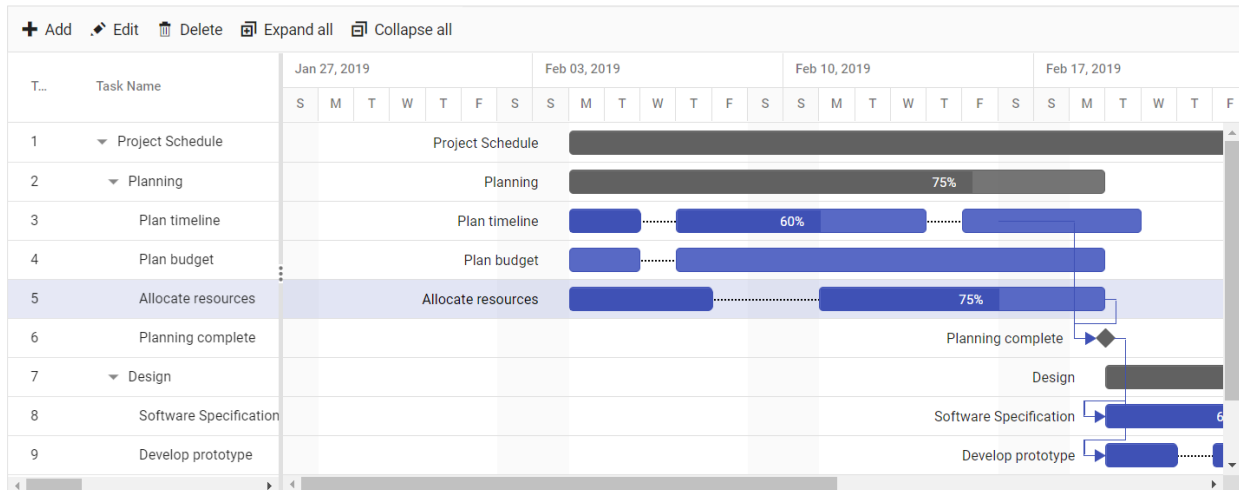
### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
 "TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
 rogress("Progress").Child("SubTasks").Segments("Segments")
).Render()
```

### SPLIT-TASK.CS

```
public ActionResult SplitTasks()
```

```
{
 ViewBag.DataSource = GanttData.SplitTasksData();
 return View();
}
```



### Self-referential

We can also define segment details as a flat data and this collection can be mapped by using [segmentData](#) property. The segment id field of this collection is mapped by using the [taskFields.SegmentId](#) property.

```
`html
```

```
GanttSegment Record1 = new GanttSegment()
```

```
{
```

```
segmentId = 2,
```

```
Duration = 2,
```

```
StartDate = new DateTime(2019, 04, 02),
```

```
};
```

```
,
```

### CSHTML

```
@(Html.EJS().Gantt("SplitTasks").DataSource((IEnumerable<object>)ViewBag.DataSource).SegmentData((IEnumerable<object>)ViewBag.Segment)
 .TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
)
 .Dependency("Predecessor").SegmentId("segmentId").Child("SubTasks"))
 .Render())
```

### SPLIT-SELFREFERENCE.CS

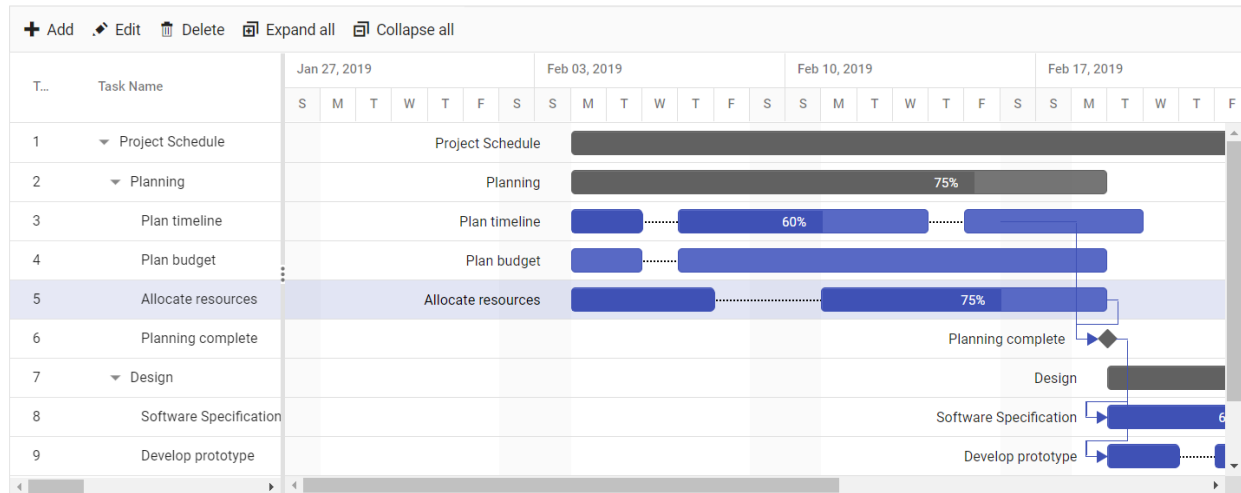
```
public ActionResult SplitTasks()
```



```

{
 ViewBag.DataSource = GanttData.SplitTasksData();
 ViewBag.Segment = GanttData.SegmentData();
 return View();
}

```



**Note:** Segment id field contains id of a task which should be split at load time.

### Improve performance by disabling validations

The [autoCalculateDateScheduling](#) property can help you reduce the time taken for the Gantt chart to render on the initial load. When this API is enabled, parent-child validation, data validation, and predecessor validation are restricted, allowing the Gantt chart to load more quickly. Since we are disabling the validations, data source provided to gantt should have all data such as start date, end date, duration, as proper data.

### CSHTML

```

@(Html.EJS().Gantt("VirtualScroll").DataSource((IEnumerable<object>)ViewBag.
DataSource).Height("450px").EnableVirtualization(true).HighlightWeekends(tru
e).AllowSelection(true)
 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress")
 .Child("SubTasks").ParentID("ParentID"))
 .LabelSettings(ls=>ls.TaskLabel("Progress")) .EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true).AllowTaskbarEdit
ing(true).ShowDeleteConfirmDialog(true))
 .AutoCalculateDateScheduling(false).Toolbar(new List<string>()
{ "Add", "Edit", "Update", "Delete", "Cancel", "ExpandAll", "CollapseAll",
"Indent", "Outdent" }).Columns(col =>
{
 col.Field("TaskId").Width(140).Add();
 col.Field("TaskName").Width(250).Add();
 col.Field("StartDate").Add();
 col.Field("EndDate").Add();
 col.Field("Duration").Add();
 col.Field("Progress").Add();
}).SplitterSettings(ss => ss.ColumnIndex(2))

```

```

 .Render()
)

```

### **VIRTUAL-SCROLL.CS**

```

public IActionResult Index()
{
 ViewBag.dataSource = GanttData.VirtualData();
 return View();
}

```

### Limitations

Gantt has the support for both Hierarchical and Self-Referential data binding. When rendering the Gantt control with SQL database, we suggest you use the Self-Referential data binding to maintain the parent-child relation. Because the complex json structure is very difficult to manage in SQL tables, we need to write complex queries, and we have to write a complex algorithm to find out the proper record details while updating/deleting the inner level task in the Gantt data source. We cannot implement both data binding for Gantt control, and this is not a recommended way. If both child and parentID are mapped, the records will not render properly because when the task id of a record defined in the hierarchy structure is assigned to the parent id of another record, in such case, the records will not properly render. As the self-referential will search the record with a particular id in flat data only, not in the inner level of records. If we map the parentID field, it will be prioritized and Gantt will be rendered based on the parentID values.

### Columns in in gantt control

The column displays information from a bound data source, and you can edit the values of column to update the task details through TreeGrid. The operations such as sorting, filtering, and searching can be performed based on column definitions. To display a Gantt column, the [Field](#) property should be mapped from the data source to the column.

---

**Note:** If the column [Field](#) is not specified in the data source, the column values will be empty.

---

The [TreeColumnIndex](#) property is used to define the expander column in the Gantt control to expand and collapse the child rows.

### Defining columns

Using the [Columns](#) property, you can define the columns in Gantt. If the columns are not defined, then the default columns will be rendered based on the mapped data source fields in the [TaskFields](#) property. Refer to the following code example for defining the columns in Gantt along with their widths.

### **CSHTML**

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration")
.Progress("Progress").Child("SubTasks")
).Columns(col =>
{
 col.Field("TaskId").Width(150).Add();
 col.Field("TaskName").HeaderText("Job Name").Width(250).Add();
}
)

```

```
}).Render()
```

## DEFININGCOLUMNS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
```

```

};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50
};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

![Alt text](images/definingColumns.png)

### Custom column header

The column header text can be defined using the [HeaderText](#) property, and you can customize the column headers using the [HeaderTemplate](#) property.

### CSHTML

```

@ (Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress(
 "Progress").Child("SubTasks")).Columns(col =>
 {

col.Field("TaskName").HeaderTemplate("#projectName").Width(150).Add();

col.Field("StartDate").HeaderTemplate("#dateTemplate").Width(150).Add();

col.Field("Duration").HeaderTemplate("#durationTemplate").Width(150).Add();

```

```

col.Field("Progress").HeaderTemplate("#progressTemplate").Width(150).Add();
 }).Render()

 <script type="text/x-template" id="projectName">
 <div>
 <div>
 Task Name
 </div>
 </div>
 </script>

 <script type="text/x-template" id="dateTemplate">
 <div>
 <div>
 Start Date
 </div>
 </div>
 </script>

 <script type="text/x-template" id="durationTemplate">
 <div>
 <div>
 Duration
 </div>
 </div>
 </script>

 <script type="text/x-template" id="progressTemplate">
 <div>
 <div>
 Progress
 </div>
 </div>
 </script>

```

### HEADERTEMPLATE.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

![Alt text](images/headerTemplate.png)

### Format

To format the cell values based on a specific culture, use the [Columns.Format](#) property. The Gantt control uses the [Internationalization](#) library to format [number](#) and [date](#) values.

**CSHTML**

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress(
 "Progress").Child("SubTasks").Columns(col =>
 {
 col.Field("TaskId").Width(150).Add();
 col.Field("Progress").Format("C").Width(150).Add();
 }).Render()
```

**FORMATCOLUMN.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![[Alt text]](images/formatColumn.png)

**Note:** By default, the [number](#) and [date](#) values are formatted in **en-US** culture.

*Number formatting*

The number or integer values can be formatted using the following format strings.

Format	Description	Remarks
----- ----- ----		
N	Denotes numeric type.   The numeric format is followed by an integer value like N2 or N3, which denotes the number of precisions to be allowed.	
C	Denotes currency type.   The currency format is followed by an integer value like C2 or C3, which denotes the number of precisions to be allowed.	
P	Denotes percentage type   The percentage format expects the input value to be in the range of 0 to 100. For example, the cell value 0.2 is formatted as 20%. The percentage format is followed by an integer value like P2, P3, which denotes the number of precisions to be allowed.	

*Date formatting*

You can format date values either using the built-in date format string or a custom format string.

For the built-in date format, you can specify the [Columns.Format](#) property as string (example: **yMd**).

You can also use the custom format string to format the date values. Some of the custom formats and the formatted date values are given in the following table.

Format	Formatted value
----- -----	
{ type:'date', format:'dd/MM/yyyy' }	04/07/2019

```
|{ type:'date', format:'dd.MM.yyyy' } | 04.07.2019|
```

```
|{ type:'date', skeleton:'short' } | 7/4/19|
```

```
|{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/2019 12:00 AM|
```

```
|{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/2019 12:00:00 AM|
```

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Columns(col =>
{
 col.Field("TaskId").Width(50).Add();
 col.Field("TaskName").Add();
 col.Field("StartDate").Format("yMd").Add();
 col.Field("Duration").Add();
 col.Field("Progress").Format("C").Add();
})
.Render()
```

### FORMATCOLUMNS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Change tree/expander column

The tree/expander column is a column in the Gantt control, that has icons to expand or collapse the parent records. You can define the tree column index in the Gantt control by using the [TreeColumnIndex](#) property and the default value of this property is 0. The following code example shows how to use this property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TreeColumnIndex(2).TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Render()
```

### TREECOLUMNINDEX.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/treeColumnIndex.png)

### Show or hide columns dynamically

You can show or hide gantt columns dynamically using external buttons by invoking the `showColumn` or `hideColumn` method. The **Progress** column is hidden and shown on button clicking.

#### CSHTML

```
@Html.EJS().Button("show").Content("Show").CssClass("e-primary").Render()
@Html.EJS().Button("hide").Content("Hide").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
.StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
.Child("SubTasks")).EditSettings(es =>
 es.AllowEditing(true)
 .Columns(col =>
 {
 col.Field("TaskId").Add();
 col.Field("TaskName").HeaderText("TaskName").Width(180).Add();
 col.Field("EndDate").HeaderText("EndDate").Width(160).Add();
 col.Field("Duration").Width(100).Add();
 col.Field("Progress").HeaderText("Progress").Add();
 })
 .Render()

<script>
 document.getElementById('show').addEventListener('click', function
(args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.showColumn(["Progress"]);
 });
 document.getElementById('hide').addEventListener('click', function
(args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.hideColumn(["Progress"]);
 });
</script>
```

#### SHOWHIDE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/showhidecol.png)



### Controlling gantt column actions

You can enable or disable gantt action for a particular column by setting the `allowFiltering`, `allowSorting`, `allowReordering`, and [allowEditing](#) properties.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
 .StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
 .Child("SubTasks")).EditSettings(es =>
 {
 es.AllowEditing(true)
 .AllowFiltering(true).AllowSorting(true).AllowReordering(true)
 .Columns(col =>
 {
 col.Field("TaskId").Add();
 col.Field("TaskName").HeaderText("TaskName")
 .AllowSorting(false).Width(180).Add();

 col.Field("StartDate").HeaderText("StartDate").AllowEditing(false).Width(160)
 .Add();
 col.Field("Duration").AllowFiltering(false).Width(100).Add();
 col.Field("Progress").AllowReordering(false).Width(100).Add();
 })
 }).Render()
```

#### GRIDACTIONS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Column type

Column type can be specified using the `columns.type` property. It specifies the type of data the column binds.

If the `format` is defined for a column, the column uses `type` to select the appropriate format option **number** or **date**.

Gantt column supports the following types:

- string
- number
- boolean
- date
- date-time

**Note:** If the `type` is not defined, it will be determined from the first record of the `dataSource`. In case if the first record of the `dataSource` is null/blank value for a column then it is necessary to define the `type` for that column.

## Timeline

In the Gantt control, timeline is used to represent the project duration as individual cells with defined unit and formats.

### Timeline view modes

Gantt contains the following in-built timeline view modes:

- Hour
- Week
- Month
- Year

Timescale mode in Gantt can be defined by using [TimelineViewMode](#) property and also we can define timescale mode of top tier and bottom tier by using [TopTier.Unit](#) and [BottomTier.Unit](#) properties.

### Week timeline mode

In the **Week** timeline mode, the upper part of the schedule header displays the weeks, whereas the bottom half of the header displays the days. Refer to the following code example.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
rogress("Progress").Child("SubTasks")).TimelineSettings(ts=>
ts.TimelineViewMode(Syncfusion.EJ2.Gantt.TimelineViewMode.Week)).Render()
```

#### WEEKMODE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/weekMode.png)

### Month timeline mode

In the **Month** timeline mode, the upper part of the schedule header displays the months, whereas the bottom header of the schedule displays its corresponding weeks. Refer to the following code example.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
rogress("Progress").Child("SubTasks")).TimelineSettings(ts=>
```

```
ts.TimelineViewMode(Syncfusion.EJ2.Gantt.TimelineViewMode.Month)).Render()
```

### **MONTHMODE.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/monthMode.png)

#### *Year timeline mode*

In the **Year** timeline mode, the upper schedule header displays the years whereas, the bottom header displays its corresponding months. Refer to the following code example.

### **CSHTML**

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate")
.EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks"))
.TimelineSettings(ts=>ts.TimelineViewMode(Syncfusion.EJ2.Gantt.TimelineViewMode.Year)).Render()
```

### **YEARMODE.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/yearMode.png)

#### *Day timeline mode*

In the **Day** timeline mode, the upper part of the header displays the days whereas, the bottom schedule header displays its corresponding hours. Refer to the following code example.

### **CSHTML**

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").DateFormat("M/d/yyyy hh:mm:ss tt").DurationUnit(Syncfusion.EJ2.Gantt.DurationUnit.Hour)
.TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate")
.Duration("Duration").Progress("Progress").Child("SubTasks")).TimelineSettings(ts=>
```

```
ts.TimelineViewMode(Syncfusion.EJ2.Gantt.TimelineViewMode.Day)).Render()
```

### DAYMODE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 5,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 5,
 Progress = 50,
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 5,
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
```

```

 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 02, 10, 10, 0),
 Duration = 5,
 Progress = 70,
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 02, 10, 10, 0),
 Duration = 5,
 Progress = 50,
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

![Alt text](images/dayMode.png)

#### Hour timeline mode

An **Hour** timeline mode tracks the tasks in minutes scale. In this mode, the upper schedule header displays hour scale and the lower schedule header displays its corresponding minutes.

#### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Height("450px").DateFormat("M/d/yyyy hh:mm:ss tt").DurationUnit(Syncfusion.EJ2.Gantt.DurationUnit.Minute)
 .TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate")
 .Duration("Duration").Progress("Progress").Child("SubTasks"))
 .TimelineSettings(ts => ts.TimelineViewMode(Syncfusion.EJ2.Gantt.TimelineViewMode.Hour))
 .Render()

```

**HOURLMODE.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 5,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 5,
 Progress = 50,
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 5,
 Progress = 50,
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
```

```

 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 02, 8, 10, 0),
 Duration = 5,
 Progress = 70,
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 02, 8, 10, 0),
 Duration = 5,
 Progress = 50,
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

![[Alt text](images/hourMode.png)]

### Timeline cells tooltip

In the Gantt control, you can enable or disable the mouse hover tooltip of timeline cells using the [TimelineSettings.ShowTooltip](#) property. The default value of this property is true. The following code example shows how to enable the timeline cells tooltip in Gantt.

### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
.StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
.Child("SubTasks")).TimelineSettings(tl => tl.ShowTooltip(true)).Render()

```

### TIMELINECELLTOOLTIP.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

![Alt text](images/timelinecellTooltip.png)

## Splitter in ASP.NET MVC Gantt component

### Splitter

In the Gantt control, the Splitter separates the TreeGrid section from the Chart section. You can change the position of the Splitter when loading the Gantt control using the [SplitterSettings](#) property. By splitting the TreeGrid from the chart, the width of the TreeGrid and chart sections will vary in the control. The [SplitterSettings.Position](#) property denotes the percentage of the TreeGrid section's width to be rendered and this property supports both pixels and percentage values. You can define the splitter position as column index value using the [SplitterSettings.ColumnIndex](#) property. You can also define the splitter position with built-in splitter view modes by using the [SplitterSettings.View](#) property. The following list is the possible values for this property:

- **Default**: Shows Grid side and Gantt side.
- **Grid**: Shows Grid side alone in Gantt.
- **Chart**: Shows chart side alone in Gantt.

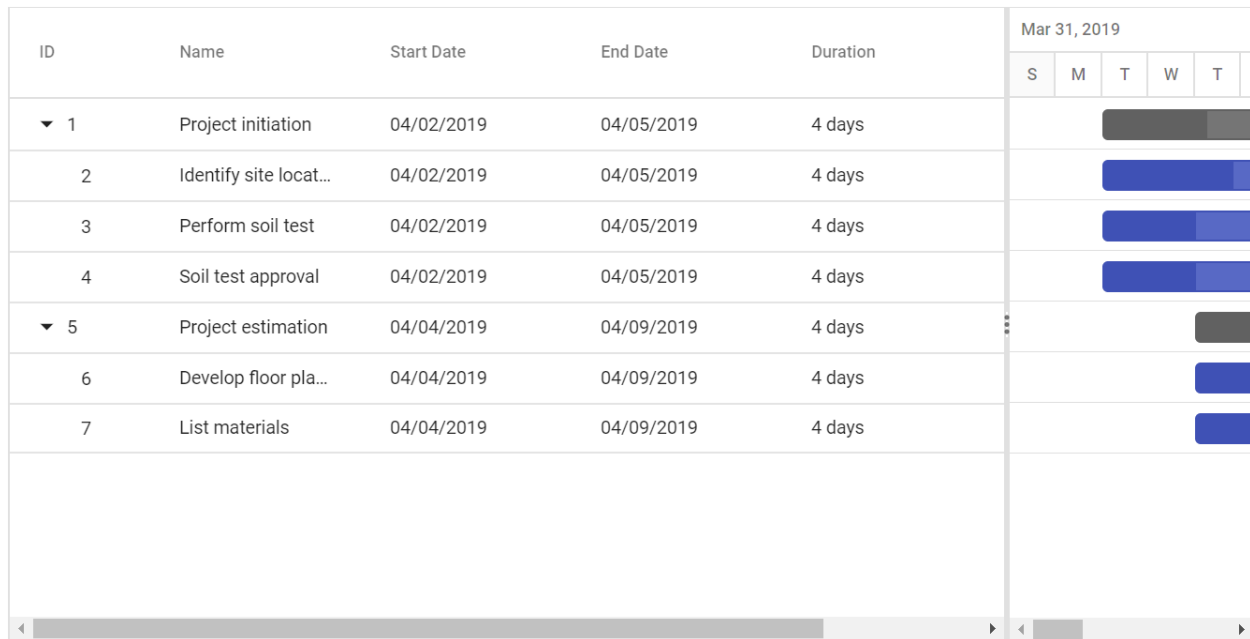
### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource).Height("450px").TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).SplitterSettings(ss=>ss.Position("80%")).Render()
```

### CHANGESPLITTER.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```





### Change splitter position dynamically

In Gantt, we can change the splitter position dynamically by using `setSplitterPosition` method. We can change the splitter position by passing value and type parameter to `setSplitterPosition` method. Type parameter will accept one of the following values 'position', 'columnIndex', 'viewType'. The following code example shows how to use this method.

### CSHTML

```
@Html.EJS().DropDownList("SplitterView").DataSource((IEnumerable<object>)ViewBag.dropdata).Width("250px").Value("Default").Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "id", Value = "view" }).Change("onViewChange").Render()
 @Html.EJS().Button("changeByPosition").Content("Change Splitter By Position").CssClass("e-primary").Render()
 @Html.EJS().Button("changeByIndex").Content("Change Splitter By ColumnIndex").CssClass("e-primary").Render()

@(Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts =>

 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Render())

<script>
 function onViewChange(e) {
 var viewType = e.value;
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.setSplitterPosition(viewType, 'view');
 }
}
```

```

document.getElementById("changeByPosition").addEventListener('click', () =>
{
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.setSplitterPosition('50%', 'position');
});
document.getElementById("changeByIndex").addEventListener('click',
() => {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.setSplitterPosition(0, 'columnIndex');
});
</script>

```

### SETSPLITTERPOSITION.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 ViewBag.dropdata = new List<object>() {
 new { id= "Default", view= "Default" },
 new { id= "Grid", view= "Grid" },
 new { id= "Chart", view= "Chart" }
 };
 return View();
}

```

### Task Scheduling

The Gantt provides support for automatic and manual task scheduling modes. It is used to indicate whether the start date and end date of all the tasks will be automatically validated or not. [taskMode](#) is the property used to change the schedule mode of a task.

The Gantt control supports three types of mode. They are:

- **Auto**: All the tasks are automatically validate.
- **Manual**: All the tasks are manually validate by the user.
- **Custom**: Both Auto and Manual tasks are render by mapped from data source.

---

**Note:** The default value of [taskMode](#) is **Auto**.

---

#### Automatically scheduled tasks

When the [taskMode](#) property is set as **Auto**, the start date and end date of all the tasks in the project will be automatically validated. That is, dates are validated based on various factors such as working time, holidays, weekends and predecessors.

### CSHTML

```

@(Html.EJS().Gantt("ResourceAllocation").DataSource((IEnumerable<object>
)ViewBag.DataSource).Height("450px").HighlightWeekends(true).AllowSelection(
true)

```

```

 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
 .Dependency("Predecessor").Child("SubTasks"))
 .TreeColumnIndex(1).TaskMode(Syncfusion.EJ2.Gantt.ScheduleMode.Auto)
 .EditSettings(es =>
es.AllowEditing(true).AllowDeleting(true).AllowTaskbarEditing(true).ShowDeleteConfirmDialog(true))
 .Toolbar(new List<string>
() { "Add", "Edit", "Update", "Delete", "Cancel", "ExpandAll", "CollapseAll" })
 .Render()
)

```

### AUTO-TASKS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = TaskModeData();
 return View();
 }
 public static List<GanttDataSource> TaskModeData()
 {
 List<GanttDataSource> GanttDataSourceCollection = new List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Parent Task 1",
 StartDate = new DateTime(2017, 02, 27),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 02, 27),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40
 };
 GanttDataSource Record1Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Child Task 2",
 };
 }
 }
}

```

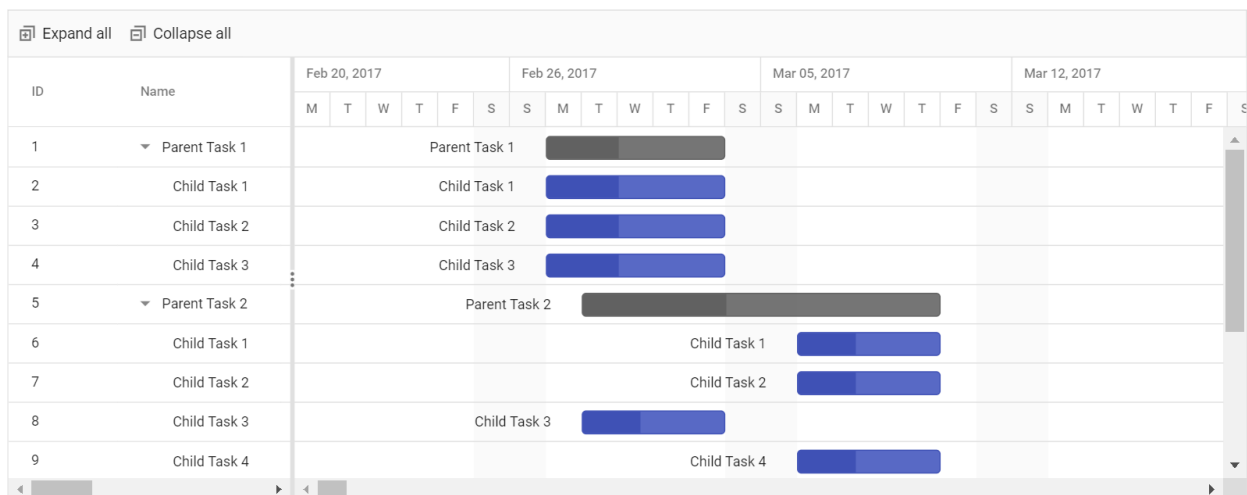
```
 StartDate = new DateTime(2017, 02, 26),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record1Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 02, 27),
 Duration = 5,
 Progress = 40,
 EndDate = new DateTime(2017, 03, 03)
 };
 Record1.SubTasks.Add(Record1Child1);
 Record1.SubTasks.Add(Record1Child2);
 Record1.SubTasks.Add(Record1Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Parent Task 2",
 StartDate = new DateTime(2017, 03, 05),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record2Child1 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 03, 06),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 };
 GanttDataSource Record2Child2 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 03, 06),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40
 };
 GanttDataSource Record2Child3 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 02, 28),
 EndDate = new DateTime(2017, 03, 05),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record2Child4 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Child Task 4",
 StartDate = new DateTime(2017, 03, 04),
```

```
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 IsManual = true
 };
 Record2.SubTasks.Add(Record2Child1);
 Record2.SubTasks.Add(Record2Child2);
 Record2.SubTasks.Add(Record2Child3);
 Record2.SubTasks.Add(Record2Child4);
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 10,
 TaskName = "Parent Task 3",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record3Child1 = new GanttDataSource()
 {
 TaskId = 11,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child2 = new GanttDataSource()
 {
 TaskId = 12,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child3 = new GanttDataSource()
 {
 TaskId = 13,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child4 = new GanttDataSource()
 {
 TaskId = 14,
 TaskName = "Child Task 4",
 StartDate = new DateTime(2017, 03, 12),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record3Child5 = new GanttDataSource()
 {
 TaskId = 15,
 TaskName = "Child Task 5",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
```

```

 Progress = 40,
 };
 Record3.SubTasks.Add(Record3Child1);
 Record3.SubTasks.Add(Record3Child2);
 Record3.SubTasks.Add(Record3Child3);
 Record3.SubTasks.Add(Record3Child4);
 Record3.SubTasks.Add(Record3Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
}
}
}

```



### Manually scheduled tasks

When the [taskMode](#) property is set as **Manual**, the start date and end date of all the tasks in the project will be same as given in the data source. That is, dates are not validated based on various factors such as dependencies between tasks, holidays, weekends, working time. We can restrict this mode in

predecessor validation alone. That is, we can automatically validate the dates based on predecessor values by enabling the [validateManualTasksOnLinking](#) property.

### CSHTML

```
@ (Html.EJS () .Gantt ("ResourceAllocation") .DataSource ((IEnumerable<object>
) ViewBag.DataSource) .Height ("450px") .HighlightWeekends (true) .AllowSelection (
true)
 .TaskFields (ts =>
ts.Id ("TaskId") .Name ("TaskName") .StartDate ("StartDate") .EndDate ("EndDate") .D
uration ("Duration") .Progress ("Progress")
 .Dependency ("Predecessor") .Child ("SubTasks"))
 .TreeColumnIndex (1) .TaskMode (Syncfusion.EJ2.Gantt.ScheduleMode.Manual)
 .EditSettings (es =>
es.AllowEditing (true) .AllowDeleting (true) .AllowTaskbarEditing (true) .ShowDele
teConfirmDialog (true))
 .Toolbar (new List<string>
() { "Add", "Edit", "Update", "Delete", "Cancel", "ExpandAll",
"CollapseAll" })
 .Render ()
)
```

### MANUAL-TASKS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = TaskModeData();
 return View();
 }
 public static List<GanttDataSource> TaskModeData()
 {
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Parent Task 1",
 StartDate = new DateTime(2017, 02, 27),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
```

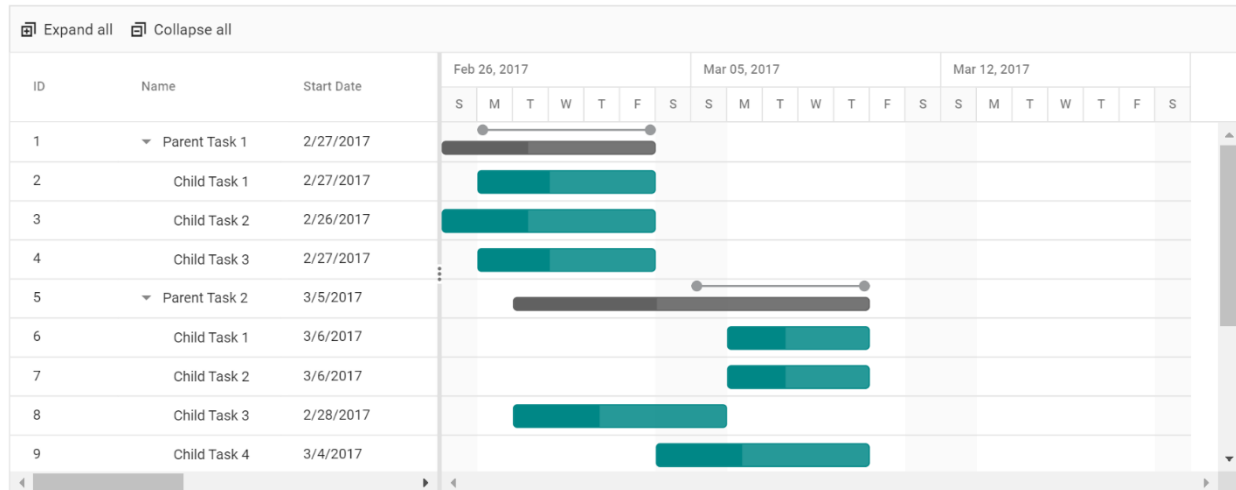
```
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 02, 27),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40
 };
 GanttDataSource Record1Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 02, 26),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record1Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 02, 27),
 Duration = 5,
 Progress = 40,
 EndDate = new DateTime(2017, 03, 03)
 };
 Record1.SubTasks.Add(Record1Child1);
 Record1.SubTasks.Add(Record1Child2);
 Record1.SubTasks.Add(Record1Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Parent Task 2",
 StartDate = new DateTime(2017, 03, 05),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record2Child1 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 03, 06),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 };
 GanttDataSource Record2Child2 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 03, 06),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40
 };
 GanttDataSource Record2Child3 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 02, 28),
```



```
 EndDate = new DateTime(2017, 03, 05),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record2Child4 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Child Task 4",
 StartDate = new DateTime(2017, 03, 04),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 IsManual = true
 };
 Record2.SubTasks.Add(Record2Child1);
 Record2.SubTasks.Add(Record2Child2);
 Record2.SubTasks.Add(Record2Child3);
 Record2.SubTasks.Add(Record2Child4);
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 10,
 TaskName = "Parent Task 3",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record3Child1 = new GanttDataSource()
 {
 TaskId = 11,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child2 = new GanttDataSource()
 {
 TaskId = 12,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child3 = new GanttDataSource()
 {
 TaskId = 13,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child4 = new GanttDataSource()
 {
 TaskId = 14,
 TaskName = "Child Task 4",
 StartDate = new DateTime(2017, 03, 12),
 EndDate = new DateTime(2017, 03, 17),
```

```
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record3Child5 = new GanttDataSource()
 {
 TaskId = 15,
 TaskName = "Child Task 5",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 };
 Record3.SubTasks.Add(Record3Child1);
 Record3.SubTasks.Add(Record3Child2);
 Record3.SubTasks.Add(Record3Child3);
 Record3.SubTasks.Add(Record3Child4);
 Record3.SubTasks.Add(Record3Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
}
}
```



### Custom

When the [taskMode](#) property is set as Custom, the scheduling mode for each tasks will be mapped from the data source field. The Boolean property [taskFields.manual](#) is used to map the manual scheduling mode field from the data source.

### CSHTML

```
@(Html.EJS().Gantt("ResourceAllocation").DataSource((IEnumerable<object>
) ViewBag.DataSource).Height("450px").HighlightWeekends(true).AllowSelection(
true)
.TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress")
.Dependency("Predecessor").Manual("IsManual").Child("SubTasks"))
.TreeColumnIndex(1).TaskMode(Syncfusion.EJ2.Gantt.ScheduleMode.Custom)
.EditSettings(es =>
es.AllowEditing(true).AllowDeleting(true).AllowTaskbarEditing(true).ShowDele
teConfirmDialog(true))
.Toolbar(new List<string>
() { "Add", "Edit", "Update", "Delete", "Cancel", "ExpandAll",
"CollapseAll" })
.Columns(col =>
{
col.Field("TaskId").Visible(false).Add();
col.Field("TaskName").HeaderText("Task Name").Add();
col.Field("IsManual").Add();
})
.Render()
)
```

### CUSTOM-TASKS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

```
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = TaskModeData();
 return View();
 }
 public static List<GanttDataSource> TaskModeData()
 {
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Parent Task 1",
 StartDate = new DateTime(2017, 02, 27),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 02, 27),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40
 };
 GanttDataSource Record1Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 02, 26),
 EndDate = new DateTime(2017, 03, 03),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record1Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 02, 27),
 Duration = 5,
 Progress = 40,
 EndDate = new DateTime(2017, 03, 03)
 };
 Record1.SubTasks.Add(Record1Child1);
 Record1.SubTasks.Add(Record1Child2);
 Record1.SubTasks.Add(Record1Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Parent Task 2",
 StartDate = new DateTime(2017, 03, 05),
```

```
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record2Child1 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Child Task 1",
 StartDate = new DateTime(2017, 03, 06),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 };
 GanttDataSource Record2Child2 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 03, 06),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40
 };
 GanttDataSource Record2Child3 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 02, 28),
 EndDate = new DateTime(2017, 03, 05),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record2Child4 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Child Task 4",
 StartDate = new DateTime(2017, 03, 04),
 EndDate = new DateTime(2017, 03, 09),
 Progress = 40,
 IsManual = true
 };
 Record2.SubTasks.Add(Record2Child1);
 Record2.SubTasks.Add(Record2Child2);
 Record2.SubTasks.Add(Record2Child3);
 Record2.SubTasks.Add(Record2Child4);
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 10,
 TaskName = "Parent Task 3",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 IsManual = true,
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record3Child1 = new GanttDataSource()
 {
 TaskId = 11,
 TaskName = "Child Task 1",
```

```

 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child2 = new GanttDataSource()
 {
 TaskId = 12,
 TaskName = "Child Task 2",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child3 = new GanttDataSource()
 {
 TaskId = 13,
 TaskName = "Child Task 3",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40
 };
 GanttDataSource Record3Child4 = new GanttDataSource()
 {
 TaskId = 14,
 TaskName = "Child Task 4",
 StartDate = new DateTime(2017, 03, 12),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 IsManual = true
 };
 GanttDataSource Record3Child5 = new GanttDataSource()
 {
 TaskId = 15,
 TaskName = "Child Task 5",
 StartDate = new DateTime(2017, 03, 13),
 EndDate = new DateTime(2017, 03, 17),
 Progress = 40,
 };
 Record3.SubTasks.Add(Record3Child1);
 Record3.SubTasks.Add(Record3Child2);
 Record3.SubTasks.Add(Record3Child3);
 Record3.SubTasks.Add(Record3Child4);
 Record3.SubTasks.Add(Record3Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 return GanttDataSourceCollection;
}

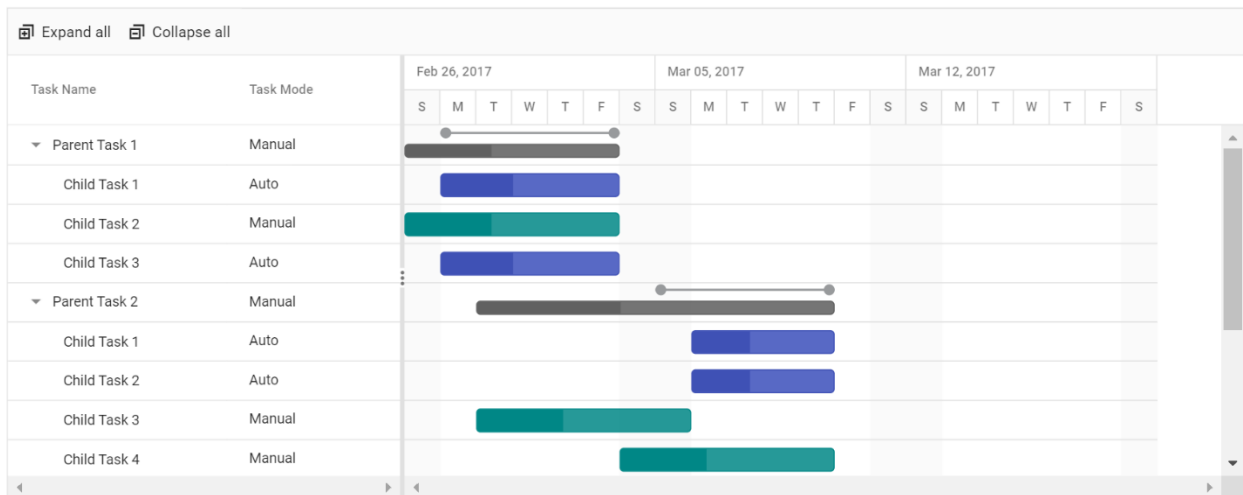
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
}

```

```

 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
 }
}

```

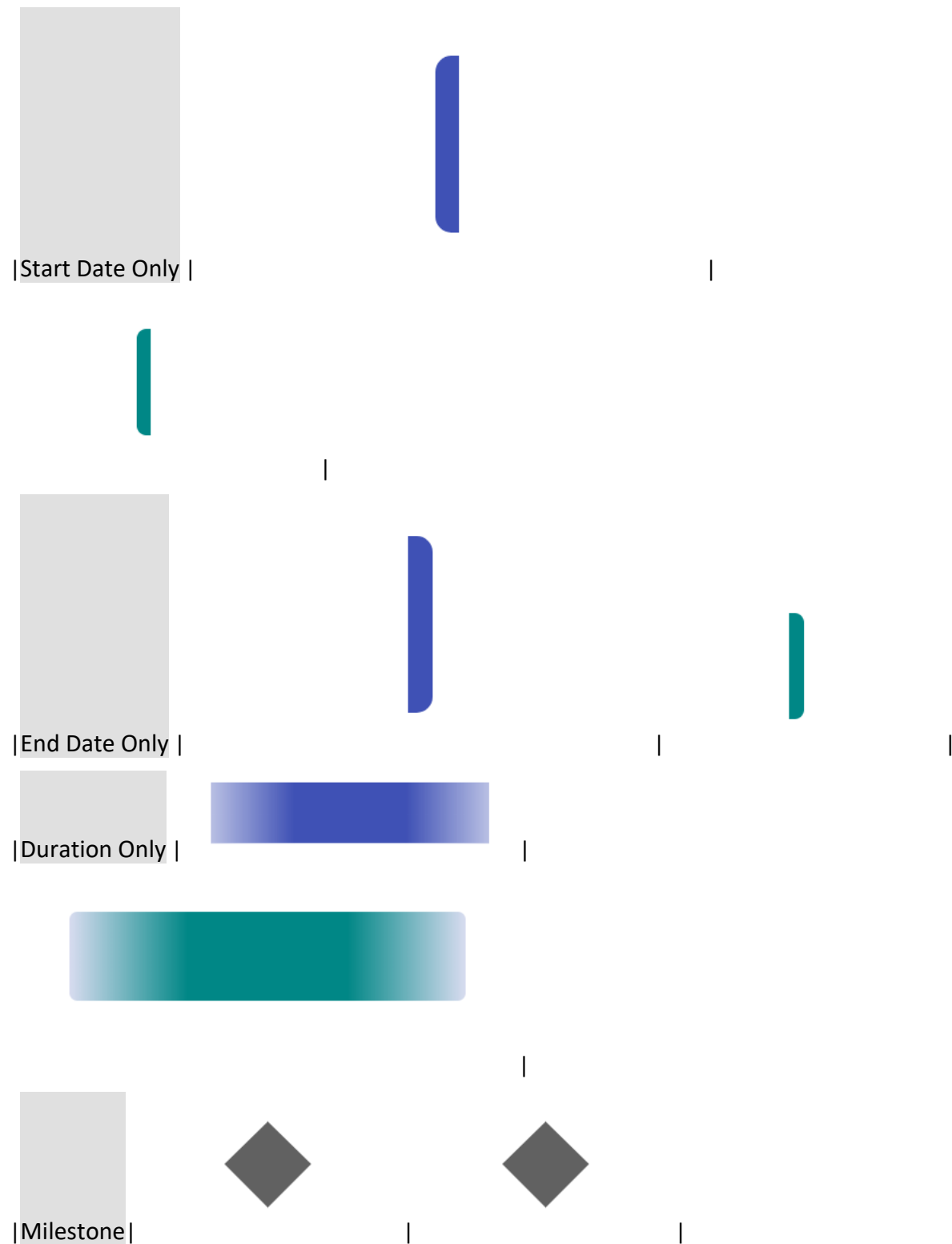


### Unscheduled tasks

Unscheduled tasks are planned for a project without any definite schedule dates. The Gantt control supports rendering the unscheduled tasks. You can create or update the tasks with anyone of start date, end date, and duration values or none. You can enable or disable the unscheduled tasks by using the [allowUnscheduledTasks](#) property. The following images represent the various types of unscheduled tasks in Gantt.

|Taskbar state |Auto |Manual|

|-----|-----|-----|



---

**Note:** A milestone is a task that has no start and end dates, but it has a duration value of zero.

---

[Define unscheduled tasks in data source](#)

You can define the various types of unscheduled tasks in the data source as follows



**CSHTML**

```
@(Html.EJS().Gantt("UnscehduledTask").DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Height("450px")
 .HighlightWeekends(true)
 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress"))
 .EditSettings(es =>
es.AllowEditing(true).AllowTaskbarEditing(true))
 .AllowUnscheduledTasks(true)
 .Render()
)
</div>
```

**UNSCHEDULED-TASKS.CS**

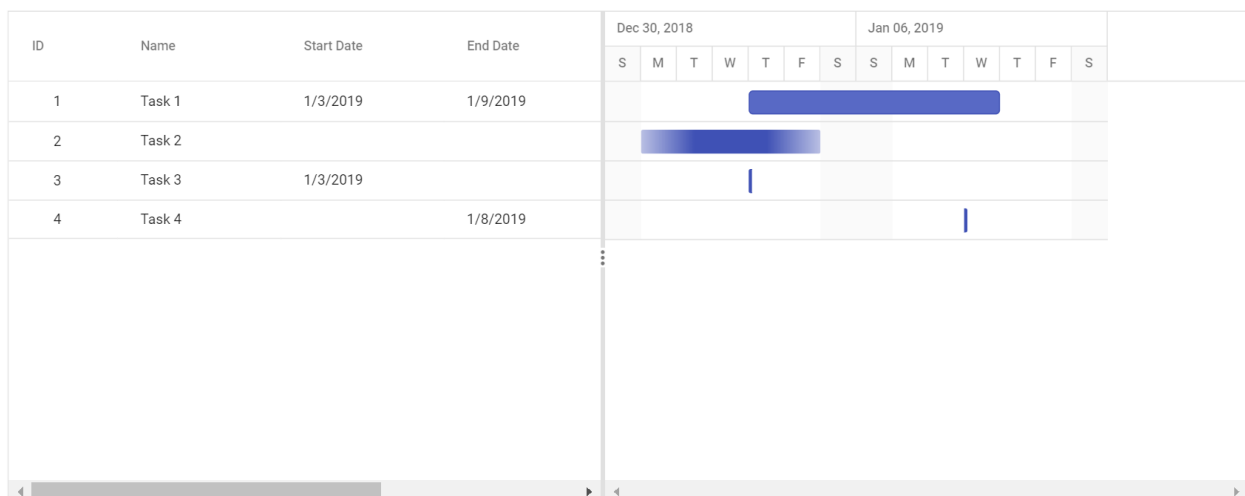
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = UnscheduledData();
 return View();
 }
 public static List<GanttDataSource> UnscheduledData()
 {
 List<GanttDataSource> ganttData = new List<GanttDataSource>();
 GanttDataSource record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Task 1",
 StartDate = new DateTime(2019, 01, 03),
 EndDate = new DateTime(2019, 01, 08),
 Duration = 5,
 TaskType = "",
 };
 GanttDataSource record2 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Task 2",
 Duration = 5,
 TaskType = "Task with duration only",
 };
 GanttDataSource record3 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Task 3",
 StartDate = new DateTime(2019, 01, 03),
```

```

 TaskType = "Task with start date only",
 };
 GanttDataSource record4 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Task 4",
 EndDate = new DateTime(2019, 01, 08),
 TaskType = "Task with end date only",
 };
 ganttData.Add(record1);
 ganttData.Add(record2);
 ganttData.Add(record3);
 ganttData.Add(record4);
 return ganttData;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
}
}

```



**Note:** If the [allowUnscheduledTasks](#) property is set to false, then the Gantt control automatically calculates the scheduled date values with a default value of duration 1 and the project start date is considered as the start date for the task.

### Working time range

In the Gantt control, working hours in a day for a project can be defined by using the [dayWorkingTime](#) property. Based on the working hours, automatic date scheduling and duration validations for a task are performed.

The following code snippet explains how to define the working time range for the project in Gantt.

### CSHTML

```
<div class="col-lg-9 control-section">
 <div class="content-wrapper">

@ (Html.EJS().Gantt("WorkingTimeRange").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px")
 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").Duration("Duration")
 .Progress("Progress").Child("SubTasks"))
 .TimelineSettings(ts => ts.TopTier(tt =>
tt.Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Day)).BottomTier(bt =>
bt.Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Hour)))
 .Render()
)
 </div>
</div>
<div class="col-lg-3 property-section">
 <table id="property" title="Properties" style="width: 100%;">
 <colgroup>
 <col style="width:55%" />
 <col style="width:45%" />
 </colgroup>
 <tr>
 <td style="width:55%">
 <div>Work Start Time</div>
 </td>
 <td style="width:45%">
 <div style="padding-top: 0px">

@ (Html.EJS().NumericTextBox("WorkStartTime").Min(0).Max(24).Value(8).Change("UpdateTime").ShowSpinButton(true).Step(0.5)
 .Render()
)
 </div>
 </td>
 </tr>
 <tr>
 <td style="width:55%">
 <div>Work End Time</div>
 </td>
 <td style="width:45%">
 <div style="padding-top: 0px">

@ (Html.EJS().NumericTextBox("WorkEndTime").Min(0).Max(24).Value(17).Change("UpdateTime").ShowSpinButton(true).Step(0.5)
 .Render()
)
 </div>
 </td>
 </tr>
 </table>
 </div>
```

```

 .Render()
)
 </div>
</td>
</tr>
</table>
</div>
<script>
 var isTimeUpdated = false;
 function UpdateTime() {
 var ganttChart =
document.getElementById("WorkingTimeRange").ej2_instances[0];
 var defaultDate = "08/08/2016", startDate = new Date(defaultDate),
endDate = new Date(defaultDate);
 var decPlace = WorkStartTime.value -
Math.floor(WorkStartTime.value);
 startDate.setHours(WorkStartTime.value);
 startDate.setMinutes(decPlace * 60);
 decPlace = WorkEndTime.value - Math.floor(WorkEndTime.value);
 endDate.setHours(WorkEndTime.value);
 endDate.setMinutes(decPlace * 60);
 /*Validate time value and update the time range*/
 if (startDate.getTime() < endDate.getTime() && isTimeUpdated ==
false) {
 var workingTime = [{ from: WorkStartTime.value, to:
WorkEndTime.value }];
 ganttChart.dayWorkingTime = workingTime;
 isTimeUpdated = false;
 }
 else {
 isTimeUpdated = true;
 WorkStartTime.value = ganttChart.dayWorkingTime[0].from;
 WorkEndTime.value =
ganttChart.dayWorkingTime[ganttChart.dayWorkingTime.length - 1].to;
 }
 isTimeUpdated = false;
 }
</script>
}

```

### WORKING-TIMERANGE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = ProjectNewData();
 return View();
 }
 }
}

```

```

public static List<GanttDataSource> ProjectNewData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 List<IndicatorsModel> Indicators = new List<IndicatorsModel>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Product concept",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child11 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Defining the product and its usage",
 StartDate = new DateTime(2019, 04, 02),
 Progress = 30,
 Duration = 3,
 };
 GanttDataSource Child12 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Defining target audience",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 3,
 };
 GanttDataSource Child13 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Prepare product sketch and notes",
 StartDate = new DateTime(2019, 04, 02),
 Progress = 30,
 Duration = 2,
 Predecessor = "2"
 };
 Record1.SubTasks.Add(Child11);
 Record1.SubTasks.Add(Child12);
 Record1.SubTasks.Add(Child13);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Concept approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 0,
 Predecessor = "3, 4",
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() { date = "04/10/2019", name=
"Design Phase", tooltip="Design phase completed", iconClass="okIcon e-icons"
}
 }
 };
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Market research",

```

```

 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record6Child1 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "Demand analysis",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record7Child1 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Customer strength",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "5",
 Progress = 30
 };
 GanttDataSource Record7Child2 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Market opportunity analysis",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "5",
 };
 Record6Child1.SubTasks.Add(Record7Child1);
 Record6Child1.SubTasks.Add(Record7Child2);
 GanttDataSource Record6Child2 = new GanttDataSource()
 {
 TaskId = 10,
 TaskName = "Competitor analysis",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "7, 8",
 Progress = 30,
 };
 GanttDataSource Record6Child3 = new GanttDataSource()
 {
 TaskId = 11,
 TaskName = "Product strength analysis",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "9",
 };
 GanttDataSource Record6Child4 = new GanttDataSource()
 {
 TaskId = 12,
 TaskName = "Research complete",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 0,
 Predecessor = "10",
 Indicators = new List<IndicatorsModel>() {

```

```

 new IndicatorsModel() { date = "04/20/2019", name=
"Research completed", tooltip="Research completed", iconClass="description
e-icons" }
 }
};
Record3.SubTasks.Add(Record6Child1);
Record3.SubTasks.Add(Record6Child2);
Record3.SubTasks.Add(Record6Child3);
Record3.SubTasks.Add(Record6Child4);
GanttDataSource Record4 = new GanttDataSource()
{
 TaskId = 13,
 TaskName = "Product design and development",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record13Child1 = new GanttDataSource()
{
 TaskId = 14,
 TaskName = "Functionality design",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 30,
 Predecessor = "12"
};
GanttDataSource Record13Child2 = new GanttDataSource()
{
 TaskId = 15,
 TaskName = "Quality design",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "12"
};
GanttDataSource Record13Child3 = new GanttDataSource()
{
 TaskId = 16,
 TaskName = "Define reliability",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Progress = 30,
 Predecessor = "15"
};
GanttDataSource Record13Child4 = new GanttDataSource()
{
 TaskId = 17,
 TaskName = "Identifying raw materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "15"
};
GanttDataSource Record13Child5 = new GanttDataSource()
{
 TaskId = 18,
 TaskName = "Define cost plan",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),

```

```
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record18Child1 = new GanttDataSource()
 {
 TaskId = 19,
 TaskName = "Manufacturing cost",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "17",
 Progress = 30
 };
 GanttDataSource Record18Child2 = new GanttDataSource()
 {
 TaskId = 20,
 TaskName = "Selling cost",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "17",
 };
 Record13Child5.SubTasks.Add(Record18Child1);
 Record13Child5.SubTasks.Add(Record18Child2);
 GanttDataSource Record13Child6 = new GanttDataSource()
 {
 TaskId = 21,
 TaskName = "Development of the final design",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record21Child1 = new GanttDataSource()
 {
 TaskId = 22,
 TaskName = "Defining dimensions and package volume",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "19, 20",
 Progress = 30
 };
 GanttDataSource Record21Child2 = new GanttDataSource()
 {
 TaskId = 23,
 TaskName = "Develop design to meet industry standards",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "22",
 };
 GanttDataSource Record21Child3 = new GanttDataSource()
 {
 TaskId = 24,
 TaskName = "Include all the details",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "23",
 };
 Record13Child6.SubTasks.Add(Record21Child1);
 Record13Child6.SubTasks.Add(Record21Child2);
 Record13Child6.SubTasks.Add(Record21Child3);
```



```

GanttDataSource Record13Child7 = new GanttDataSource()
{
 TaskId = 25,
 TaskName = "CAD Computer-aided design",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 30,
 Predecessor = "24"
};
GanttDataSource Record13Child8 = new GanttDataSource()
{
 TaskId = 26,
 TaskName = "CAM Computer-aided manufacturing",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "25"
};
GanttDataSource Record13Child9 = new GanttDataSource()
{
 TaskId = 27,
 TaskName = "Design complete",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 0,
 Predecessor = "26",
};
GanttDataSource Record5 = new GanttDataSource()
{
 TaskId = 28,
 TaskName = "Prototype testing",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 30,
 Predecessor = "27"
};
GanttDataSource Record6 = new GanttDataSource()
{
 TaskId = 29,
 TaskName = "Include feedback",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "28ss",
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() { date = "05/24/2019", name=
"Production phase", tooltip="Production phase completed", iconClass="okIcon
e-icons" }
 };
};
GanttDataSource Record7 = new GanttDataSource()
{
 TaskId = 30,
 TaskName = "Manufacturing",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 5,
 Progress = 30,
 Predecessor = "28, 29"
};
GanttDataSource Record8 = new GanttDataSource()

```

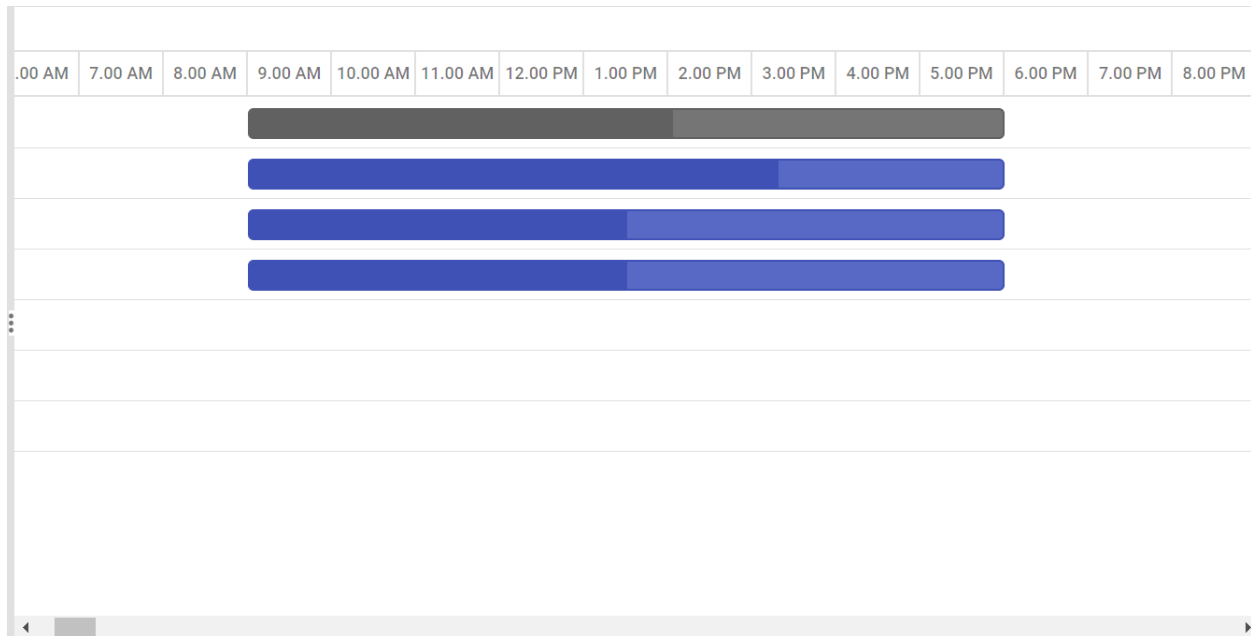
```
{
 TaskId = 31,
 TaskName = "Assembling materials to finsihed goods",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 5,
 Predecessor = "30"
};
Record4.SubTasks.Add(Record13Child1);
Record4.SubTasks.Add(Record13Child2);
Record4.SubTasks.Add(Record13Child3);
Record4.SubTasks.Add(Record13Child4);
Record4.SubTasks.Add(Record13Child5);
Record4.SubTasks.Add(Record13Child6);
Record4.SubTasks.Add(Record13Child7);
Record4.SubTasks.Add(Record13Child8);
Record4.SubTasks.Add(Record13Child9);
GanttDataSource Record9 = new GanttDataSource()
{
 TaskId = 32,
 TaskName = "Feedback and testing",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record9Child1 = new GanttDataSource()
{
 TaskId = 33,
 TaskName = "Internal testing and feedback",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 45,
 Predecessor = "31",
};
GanttDataSource Record9Child2 = new GanttDataSource()
{
 TaskId = 34,
 TaskName = "Customer testing and feedback",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Predecessor = "33",
};
Record9.SubTasks.Add(Record9Child1);
Record9.SubTasks.Add(Record9Child2);
GanttDataSource Record10 = new GanttDataSource()
{
 TaskId = 35,
 TaskName = "Final product development",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record10Child1 = new GanttDataSource()
{
 TaskId = 36,
 TaskName = "Important improvements",
 StartDate = new DateTime(2019, 04, 04),
```

```

 Duration = 4,
 Progress = 30,
 Predecessor = "34",
 };
 GanttDataSource Record10Child2 = new GanttDataSource()
 {
 TaskId = 37,
 TaskName = "Address any unforeseen issues",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 30,
 Predecessor = "36ss",
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() { date = "06/21/2019", name=
 "Sales and marketing", tooltip="Sales and marketing", iconClass="description
 e-icons" }
 }
 };
 Record10.SubTasks.Add(Record10Child1);
 Record10.SubTasks.Add(Record10Child2);
 GanttDataSource Record11 = new GanttDataSource()
 {
 TaskId = 38,
 TaskName = "Final product",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record11Child1 = new GanttDataSource()
 {
 TaskId = 39,
 TaskName = "Branding product",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "37",
 };
 GanttDataSource Record11Child2 = new GanttDataSource()
 {
 TaskId = 40,
 TaskName = "Marketing and presales",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 30,
 Predecessor = "39",
 };
 Record11.SubTasks.Add(Record11Child1);
 Record11.SubTasks.Add(Record11Child2);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 GanttDataSourceCollection.Add(Record4);
 GanttDataSourceCollection.Add(Record5);
 GanttDataSourceCollection.Add(Record6);
 GanttDataSourceCollection.Add(Record7);
 GanttDataSourceCollection.Add(Record8);
 GanttDataSourceCollection.Add(Record9);
 GanttDataSourceCollection.Add(Record10);

```

```
GanttDataSourceCollection.Add(Record11);
return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public List<ResourceModel> Resources { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
 public List<IndicatorsModel> Indicators { get; set; }
}
public class ResourceModel
{
 public int ResourceId { get; set; }
 public Nullable<int> ResourceUnit { get; set; }
}
public class IndicatorsModel
{
 public string date { get; set; }
 public string iconClass { get; set; }
 public string name { get; set; }
 public string tooltip { get; set; }
}
}
```



**Note:** \* Individual tasks can lie between any time within the defined working time range of the project.

<br/>\* The [dayWorkingTime](#) property is used to define the working time for the whole project.

#### Weekend/Non-working days

Non-working days/weekend are used to represent the non-productive days in a project. You can define the non-working days in a week using the [workWeek](#) property in Gantt.

#### CSHTML

```
@(Html.EJS().Gantt("WorkWeek").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").HighlightWeekends(true)
 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").Duration("Duration")
 .Progress("Progress").Child("SubTasks"))
 .WorkWeek(new string[] { "Sunday", "Monday", "Tuesday",
 "Wednesday", "Thursday", "Friday" })
 .Render()
)
```

#### WORKING-DAYS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = ProjectNewData();
 return View();
 }
 }
}
```

```

 }
 public static List<GanttDataSource> ProjectNewData()
 {
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 List<IndicatorsModel> Indicators = new List<IndicatorsModel>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Product concept",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child11 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Defining the product and its usage",
 StartDate = new DateTime(2019, 04, 02),
 Progress = 30,
 Duration = 3,
 };
 GanttDataSource Child12 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Defining target audience",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 3,
 };
 GanttDataSource Child13 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Prepare product sketch and notes",
 StartDate = new DateTime(2019, 04, 02),
 Progress = 30,
 Duration = 2,
 Predecessor = "2"
 };
 Record1.SubTasks.Add(Child11);
 Record1.SubTasks.Add(Child12);
 Record1.SubTasks.Add(Child13);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Concept approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 0,
 Predecessor = "3, 4",
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() { date = "04/10/2019", name=
"Design Phase", tooltip="Design phase completed", iconClass="okIcon e-icons"
 }
 }
 };
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 6,

```

```

 TaskName = "Market research",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record6Child1 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "Demand analysis",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record7Child1 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Customer strength",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "5",
 Progress = 30
 };
 GanttDataSource Record7Child2 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Market opportunity analysis",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "5",
 };
 Record6Child1.SubTasks.Add(Record7Child1);
 Record6Child1.SubTasks.Add(Record7Child2);
 GanttDataSource Record6Child2 = new GanttDataSource()
 {
 TaskId = 10,
 TaskName = "Competitor analysis",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "7, 8",
 Progress = 30,
 };
 GanttDataSource Record6Child3 = new GanttDataSource()
 {
 TaskId = 11,
 TaskName = "Product strength analysis",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "9",
 };
 GanttDataSource Record6Child4 = new GanttDataSource()
 {
 TaskId = 12,
 TaskName = "Research complete",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 0,
 Predecessor = "10",
 Indicators = new List<IndicatorsModel>() {

```

```

 new IndicatorsModel() { date = "04/20/2019", name=
"Research completed", tooltip="Research completed", iconClass="description
e-icons" }
 }
};
Record3.SubTasks.Add(Record6Child1);
Record3.SubTasks.Add(Record6Child2);
Record3.SubTasks.Add(Record6Child3);
Record3.SubTasks.Add(Record6Child4);
GanttDataSource Record4 = new GanttDataSource()
{
 TaskId = 13,
 TaskName = "Product design and development",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record13Child1 = new GanttDataSource()
{
 TaskId = 14,
 TaskName = "Functionality design",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 30,
 Predecessor = "12"
};
GanttDataSource Record13Child2 = new GanttDataSource()
{
 TaskId = 15,
 TaskName = "Quality design",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "12"
};
GanttDataSource Record13Child3 = new GanttDataSource()
{
 TaskId = 16,
 TaskName = "Define reliability",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Progress = 30,
 Predecessor = "15"
};
GanttDataSource Record13Child4 = new GanttDataSource()
{
 TaskId = 17,
 TaskName = "Identifying raw materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "15"
};
GanttDataSource Record13Child5 = new GanttDataSource()
{
 TaskId = 18,
 TaskName = "Define cost plan",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),

```



```
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record18Child1 = new GanttDataSource()
 {
 TaskId = 19,
 TaskName = "Manufacturing cost",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "17",
 Progress = 30
 };
 GanttDataSource Record18Child2 = new GanttDataSource()
 {
 TaskId = 20,
 TaskName = "Selling cost",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "17",
 };
 Record13Child5.SubTasks.Add(Record18Child1);
 Record13Child5.SubTasks.Add(Record18Child2);
 GanttDataSource Record13Child6 = new GanttDataSource()
 {
 TaskId = 21,
 TaskName = "Development of the final design",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record21Child1 = new GanttDataSource()
 {
 TaskId = 22,
 TaskName = "Defining dimensions and package volume",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "19, 20",
 Progress = 30
 };
 GanttDataSource Record21Child2 = new GanttDataSource()
 {
 TaskId = 23,
 TaskName = "Develop design to meet industry standards",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 2,
 Predecessor = "22",
 };
 GanttDataSource Record21Child3 = new GanttDataSource()
 {
 TaskId = 24,
 TaskName = "Include all the details",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "23",
 };
 Record13Child6.SubTasks.Add(Record21Child1);
 Record13Child6.SubTasks.Add(Record21Child2);
 Record13Child6.SubTasks.Add(Record21Child3);
```

```

GanttDataSource Record13Child7 = new GanttDataSource()
{
 TaskId = 25,
 TaskName = "CAD Computer-aided design",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 30,
 Predecessor = "24"
};
GanttDataSource Record13Child8 = new GanttDataSource()
{
 TaskId = 26,
 TaskName = "CAM Computer-aided manufacturing",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "25"
};
GanttDataSource Record13Child9 = new GanttDataSource()
{
 TaskId = 27,
 TaskName = "Design complete",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 0,
 Predecessor = "26",
};
GanttDataSource Record5 = new GanttDataSource()
{
 TaskId = 28,
 TaskName = "Prototype testing",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 30,
 Predecessor = "27"
};
GanttDataSource Record6 = new GanttDataSource()
{
 TaskId = 29,
 TaskName = "Include feedback",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "28ss",
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() { date = "05/24/2019", name=
"Production phase", tooltip="Production phase completed", iconClass="okIcon
e-icons" }
 };
GanttDataSource Record7 = new GanttDataSource()
{
 TaskId = 30,
 TaskName = "Manufacturing",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 5,
 Progress = 30,
 Predecessor = "28, 29"
};
GanttDataSource Record8 = new GanttDataSource()

```

```
{
 TaskId = 31,
 TaskName = "Assembling materials to finsihed goods",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 5,
 Predecessor = "30"
};
Record4.SubTasks.Add(Record13Child1);
Record4.SubTasks.Add(Record13Child2);
Record4.SubTasks.Add(Record13Child3);
Record4.SubTasks.Add(Record13Child4);
Record4.SubTasks.Add(Record13Child5);
Record4.SubTasks.Add(Record13Child6);
Record4.SubTasks.Add(Record13Child7);
Record4.SubTasks.Add(Record13Child8);
Record4.SubTasks.Add(Record13Child9);
GanttDataSource Record9 = new GanttDataSource()
{
 TaskId = 32,
 TaskName = "Feedback and testing",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record9Child1 = new GanttDataSource()
{
 TaskId = 33,
 TaskName = "Internal testing and feedback",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 45,
 Predecessor = "31",
};
GanttDataSource Record9Child2 = new GanttDataSource()
{
 TaskId = 34,
 TaskName = "Customer testing and feedback",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Predecessor = "33",
};
Record9.SubTasks.Add(Record9Child1);
Record9.SubTasks.Add(Record9Child2);
GanttDataSource Record10 = new GanttDataSource()
{
 TaskId = 35,
 TaskName = "Final product development",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record10Child1 = new GanttDataSource()
{
 TaskId = 36,
 TaskName = "Important improvements",
 StartDate = new DateTime(2019, 04, 04),
```

```

 Duration = 4,
 Progress = 30,
 Predecessor = "34",
 };
 GanttDataSource Record10Child2 = new GanttDataSource()
 {
 TaskId = 37,
 TaskName = "Address any unforeseen issues",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 30,
 Predecessor = "36ss",
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() { date = "06/21/2019", name=
 "Sales and marketing", tooltip="Sales and marketing", iconClass="description
 e-icons" }
 }
 };
 Record10.SubTasks.Add(Record10Child1);
 Record10.SubTasks.Add(Record10Child2);
 GanttDataSource Record11 = new GanttDataSource()
 {
 TaskId = 38,
 TaskName = "Final product",
 StartDate = new DateTime(2019, 04, 04),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record11Child1 = new GanttDataSource()
 {
 TaskId = 39,
 TaskName = "Branding product",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Predecessor = "37",
 };
 GanttDataSource Record11Child2 = new GanttDataSource()
 {
 TaskId = 40,
 TaskName = "Marketing and presales",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 30,
 Predecessor = "39",
 };
 Record11.SubTasks.Add(Record11Child1);
 Record11.SubTasks.Add(Record11Child2);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 GanttDataSourceCollection.Add(Record4);
 GanttDataSourceCollection.Add(Record5);
 GanttDataSourceCollection.Add(Record6);
 GanttDataSourceCollection.Add(Record7);
 GanttDataSourceCollection.Add(Record8);
 GanttDataSourceCollection.Add(Record9);
 GanttDataSourceCollection.Add(Record10);

```

```

 GanttDataSourceCollection.Add(Record11);
 return GanttDataSourceCollection;
 }
 public class GanttDataSource
 {
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public List<ResourceModel> Resources { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
 public List<IndicatorsModel> Indicators { get; set; }
 }
 public class ResourceModel
 {
 public int ResourceId { get; set; }
 public Nullable<int> ResourceUnit { get; set; }
 }
 public class IndicatorsModel
 {
 public string date { get; set; }
 public string iconClass { get; set; }
 public string name { get; set; }
 public string tooltip { get; set; }
 }
 public class CheckList
 {
 public string id { get; set; }
 public string day { get; set; }
 public static List<CheckList> Days()
 {
 List<CheckList> Data = new List<CheckList>();
 Data.Add(new CheckList { id = "Sunday", day = "Sunday" });
 Data.Add(new CheckList { id = "Monday", day = "Monday" });
 Data.Add(new CheckList { id = "Tuesday", day = "Tuesday" });
 Data.Add(new CheckList { id = "Wednesday", day = "Wednesday" });

 Data.Add(new CheckList { id = "Thursday", day = "Thursday" });

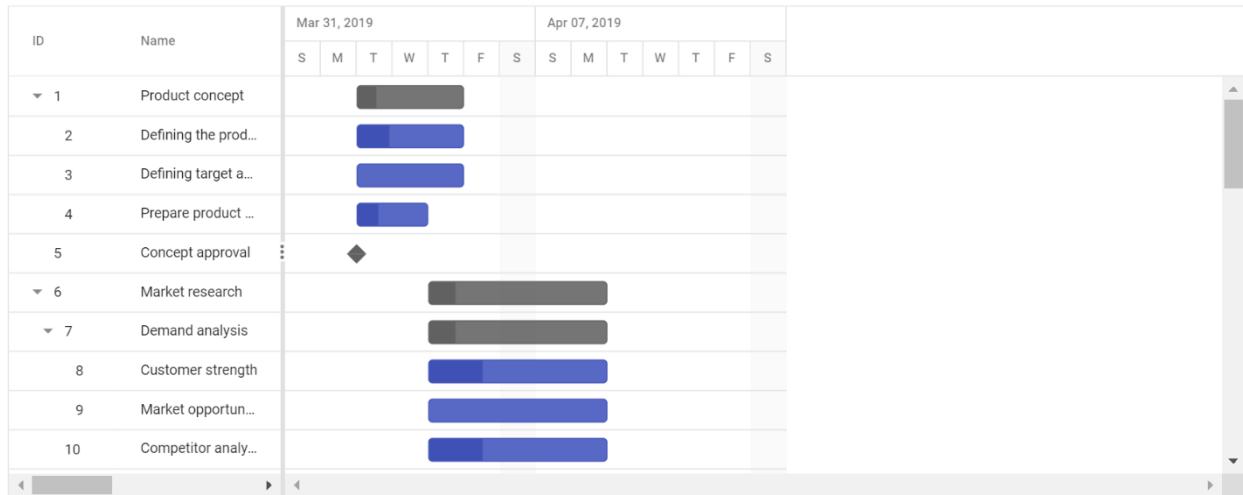
 Data.Add(new CheckList { id = "Friday", day = "Friday" });
 Data.Add(new CheckList { id = "Saturday", day = "Saturday" });

 return Data;
 }
 }
}

```

```
}

```



**Note:** By default, Saturdays and Sundays are considered as non-working days/weekend in a project.

In the Gantt control, you can make weekend as working day by setting the [includeWeekend](#) property to **true**.

### Duration units

In Gantt, the task's duration value can be measured by the following duration units,

- Day
- Hour
- Minute

In Gantt, we can define duration unit for whole project by using [durationUnit](#) property, when we defines the value for this property, this unit will be applied for all task which don't has duration unit value. And each task in the project can be defined with different duration units and the duration unit of a task can be defined by the following ways,

- Using [taskFields.durationUnit](#) property, to map the duration unit data source field.
- Defining the duration unit value along with the duration field in the data source.

### Mapping the duration unit field

The below code snippet explains the mapping of duration unit data source field to the Gantt control using the [taskFields.durationUnit](#) property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
 .StartDate("StartDate").EndDate("EndDate").Duration("Duration").DurationUnit("DurationUnit")
 .Progress("Progress").Child("SubTasks")).Render()
```

**DURATIONUNITS.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

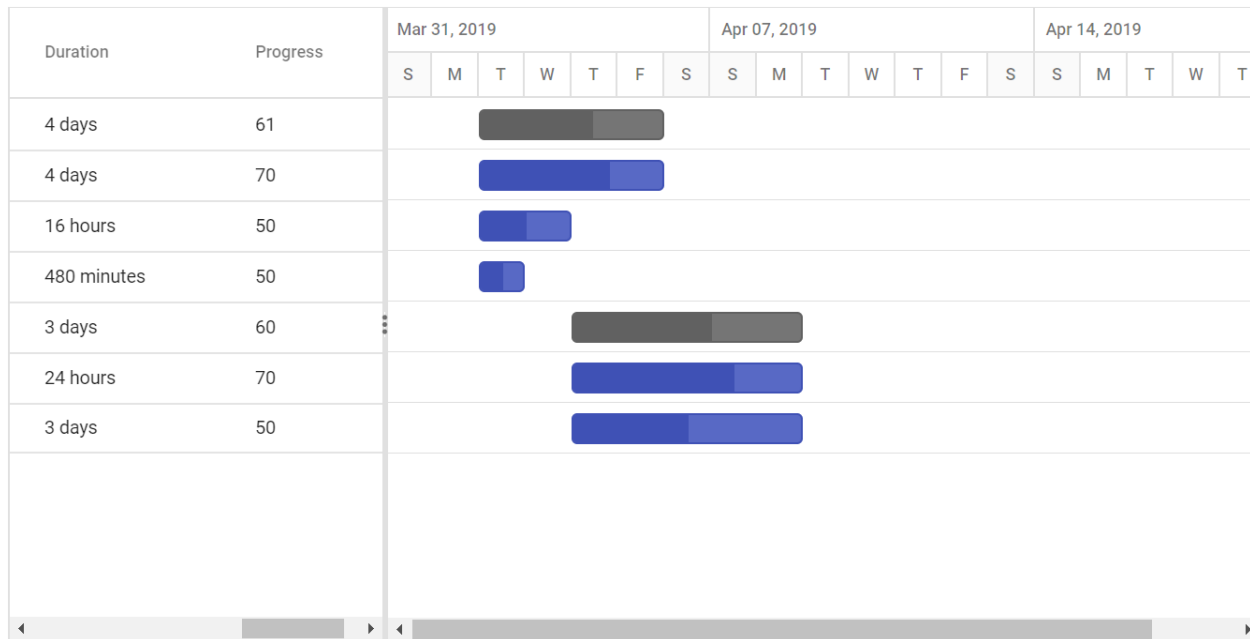
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 DurationUnit="day"
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 16,
 Progress = 50,
 DurationUnit = "hour"
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 480,
 Progress = 50,
 DurationUnit = "minute"
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
```

```
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 24,
 Progress = 70,
 DurationUnit = "hour"
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 DurationUnit = "day"
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public string DurationUnit { get; set; }

 public List<GanttDataSource> SubTasks { get; set; }
}
```





**Note:** The default value of the [durationUnit](#) property is day.

#### Defining duration unit along with duration field

Duration units for the tasks can also be defined along with the duration values, the below code snippet explains the duration unit for a task along with duration value,

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
rogress("Progress").Child("SubTasks").Render()
```

#### DURATIONUNITSWITHDURATION.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
}
```

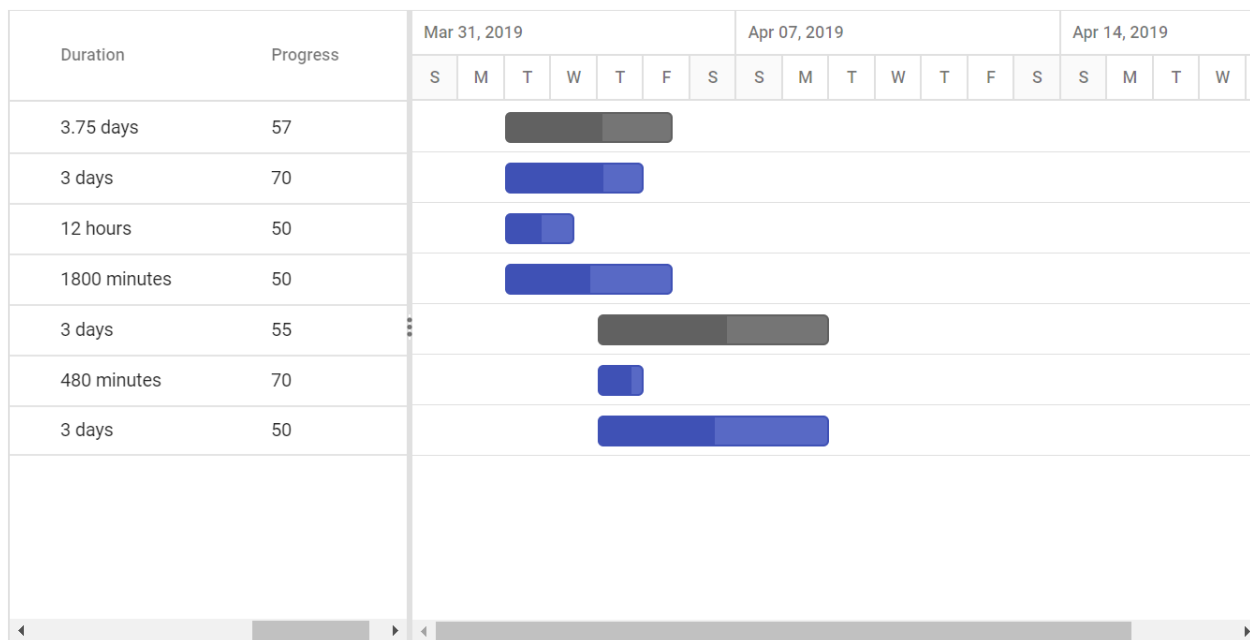
```
GanttDataSource Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = "3days",
 Progress = 70,
};
GanttDataSource Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = "12hours",
 Progress = 50
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = "1800minutes",
 Progress = 50
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = "480minutes",
 Progress = 70
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = "3days",
 Progress = 50
};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
```

```

 return GanttDataSourceCollection;
 }

 public class GanttDataSource
 {
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public string Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 }

```



**Note:** The edit type of the duration column in Gantt is string, to support editing the duration field along with duration units.

## Taskbar in ASP.NET MVC Gantt Chart Component

### Taskbar template

You can design your own taskbars to view the tasks in Gantt by using [TaskbarTemplate](#) property. And it is possible to map the template script element's ID value to this property. It is also possible to customize the parent taskbars and milestones with custom templates by using [ParentTaskbarTemplate](#) and [MilestoneTemplate](#) properties.

### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").RowHeight(60).TaskbarTemplate(
 "#TaskbarTemplate").ParentTaskbarTemplate("#ParentTaskbarTemplate").MilestoneTemplate(
 "#MilestoneTemplate").TaskFields(ts =>

```

```

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Render()

<script type="text/x-jsrender" id="TaskbarTemplate">
 <div class="e-gantt-child-taskbar-inner-div e-gantt-child-taskbar"
style="height:100%">
 <div class="e-gantt-child-progressbar-inner-div e-gantt-child-progressbar" style="width:${ganttProperties.progressWidth}px;height:100%">
 ${taskData.TaskName}
 </div>
 </div>
</script>
<script type="text/x-jsrender" id="ParentTaskbarTemplate">
 <div class="e-gantt-parent-taskbar-inner-div e-gantt-parent-taskbar"
style="height:100%">
 <div class="e-gantt-parent-progressbar-inner-div e-gantt-parent-progressbar" style="width:${ganttProperties.progressWidth}px;height:100%">
 ${taskData.TaskName}
 </div>
 </div>
</script>
<script type="text/x-jsrender" id="MilestoneTemplate">
 <div class="e-gantt-milestone" style="position:absolute;">
 <div class="e-milestone-top" style="border-right-width:15px;border-left-width:15px;border-bottom-width:15px;"></div>
 <div class="e-milestone-bottom" style="top:15px;border-right-width:15px; border-left-width:15px; border-top-width:15px;">
 </div>
 </div>
</script>

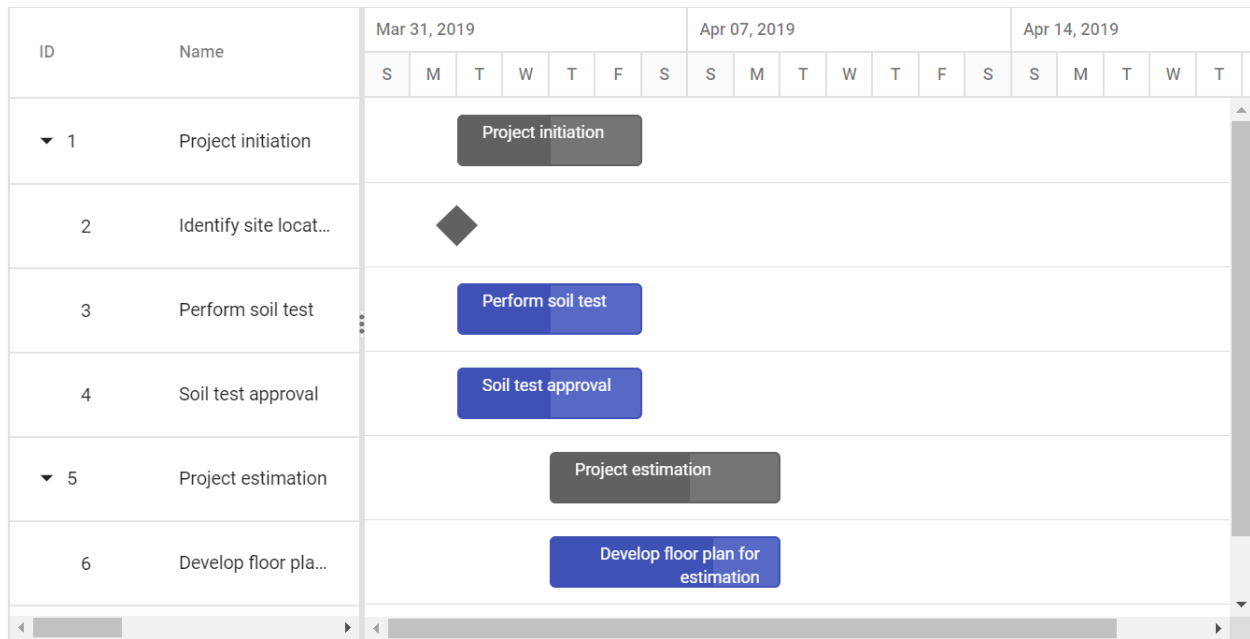
```

### TASKBARTEMPLATE.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```



### Taskbar customization

#### Taskbar Height

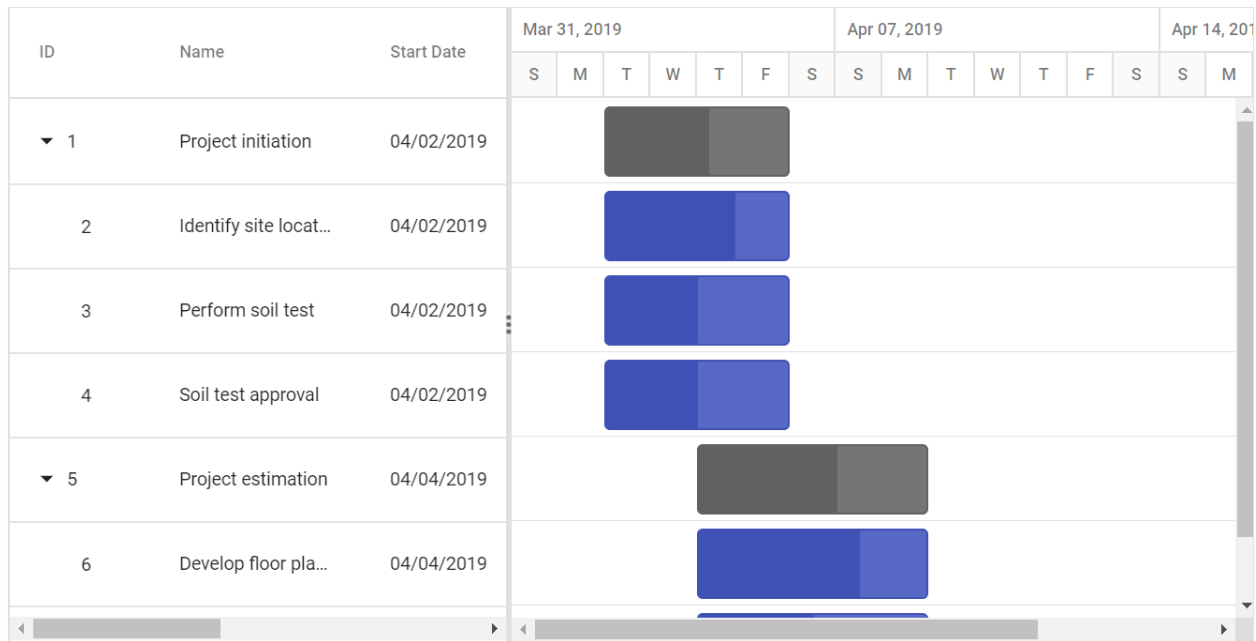
Height of child taskbars and parent taskbars can be customized by using [TaskbarHeight](#) property. The following code example shows how to use the property.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").RowHeight(60).TaskbarHeight(50).TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration(
 "Duration").Progress("Progress").Child("SubTasks")).Render()
```

#### TASKBARHEIGHT.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```



**Note:** The [TaskbarHeight](#) value should be lower than the [RowHeight](#) property value and these properties accept only pixel values.

#### Conditional formatting

The default taskbar UI can be replaced with custom templates using the [QueryTaskbarInfo](#) event. The following code example shows customizing the taskbar UI based on task progress values in the Gantt control.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").QueryTaskbarInfo("queryTaskbarInfo").TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Render()

<script>
 function queryTaskbarInfo(args) {
 if (args.data.Progress == 50) {
 args.progressBarBgColor = "red";
 } else if (args.data.Progress == 70) {
 args.progressBarBgColor = "yellow";
 } else if (args.data.Progress == 80) {
 args.progressBarBgColor = "lightgreen";
 }
 }
</script>
```

#### CONDITIONALFORMATTING.CS

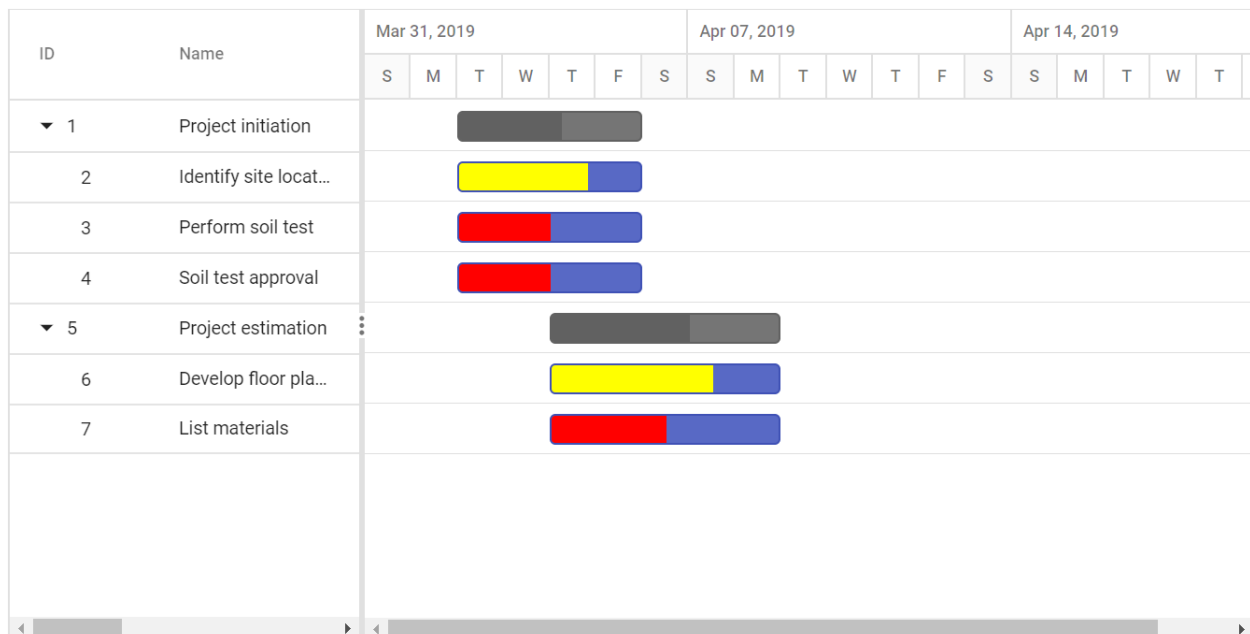
```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}
```

```
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
```

```

 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```



#### Change gripper icon in taskbar

You can change the gripper icon in the taskbar by applying styles to their respective class elements.

#### CSHTML

```

@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Editing").DataSource((IEnumerable<object>)Model).TaskFields(
 ts => ts.Id("TaskId").Name("TaskName").StartDate(
 "StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency(
 "Predecessor").Child("SubTasks").EditSettings(es => es.AllowTaskbarEditing(true)).Render()
)

```



```

<style>
/* change gripper icon */
.e-gantt .e-left-resize-gripper::before, .e-gantt .e-right-resize-
gripper::before {
 content: '\e934';
}
.e-gantt .e-left-resize-gripper, .e-gantt .e-right-resize-gripper {
 transform: rotate(90deg);
}
</style>

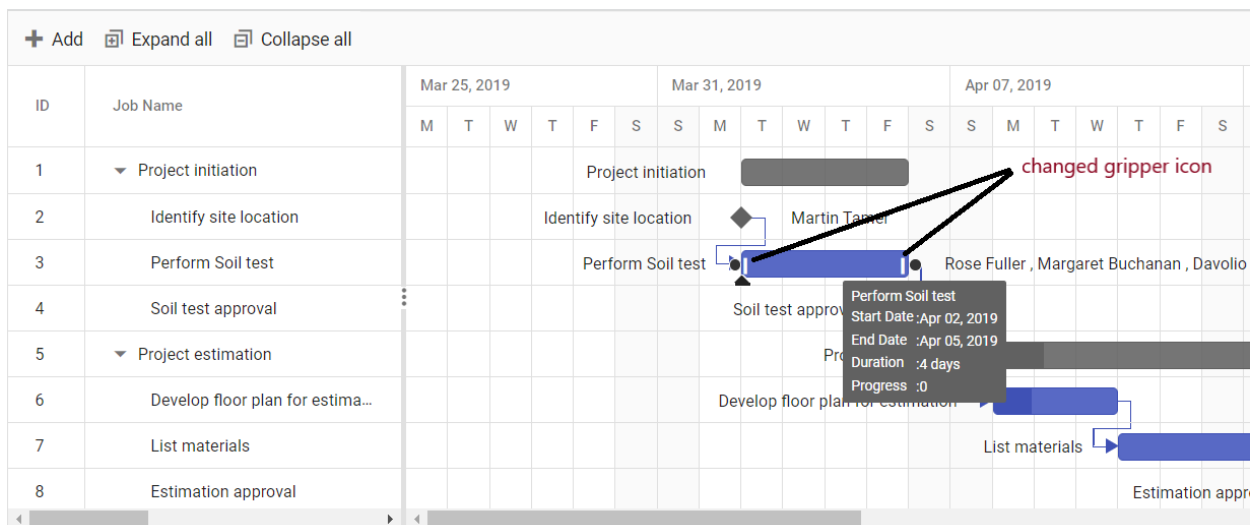
```

## DATA.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```



## Connector lines

The width and background color of connector lines in Gantt can be customized using the [ConnectorLineWidth](#) and [ConnectorLineBackground](#) properties. The following code example shows how to use these properties.

## CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").ConnectorLineWidth(3).ConnectorLineBackground("red").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks").Dependency("Dependency")).Render()

```

**DEPENDENCYFORMATTING.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

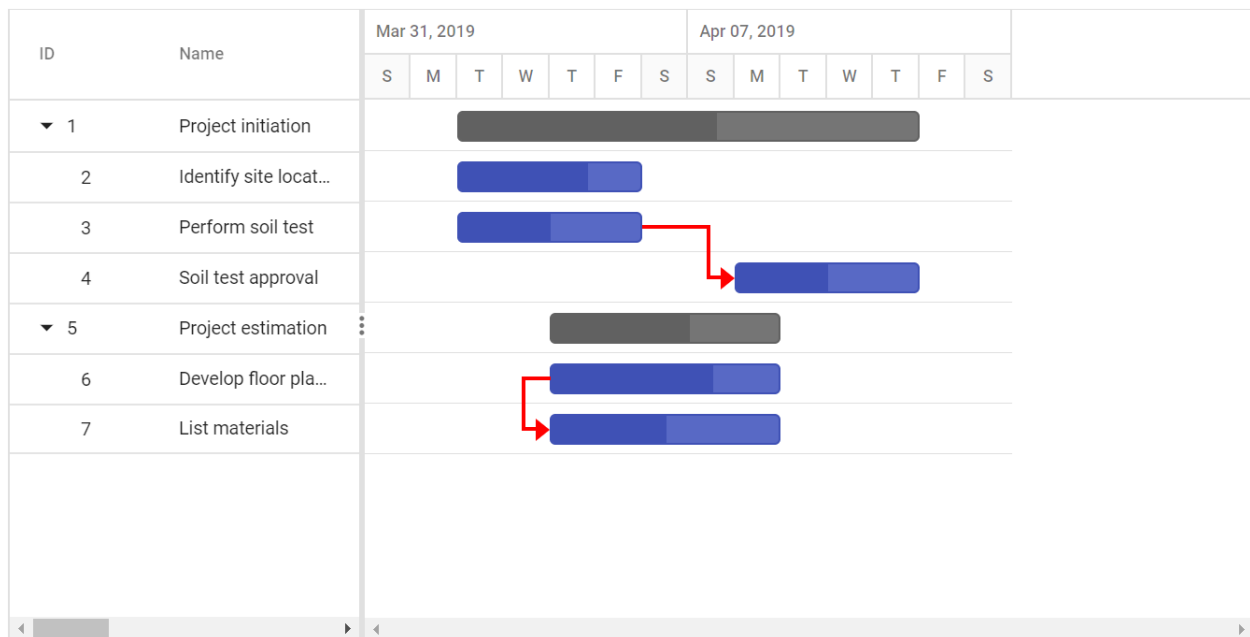
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Dependency="3FS"
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
```

```

 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Dependency = "6SS"
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string Dependency { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```



## Tooltip

In the Gantt control, you can enable or disable the mouse hover tooltip for the following UI elements using the [TooltipSettings.ShowTooltip](#) property:

- Taskbar
- Connector line
- Baseline
- Event marker

## CSHTML

```
<style>
 .e-gantt .e-gantt-chart .e-custom-event-marker {
 width: 1px;
 border-left: 2px green dotted;
 }
</style>

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.D
ataSource).Height("450px").RenderBaseline(
 true).BaselineColor("red").TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").Dependency("Predecessor").StartDate(
"StartDate").BaselineStartDate("BaselineStartDate").BaselineEndDate("Baselin
eEndDate").Duration("Duration"
).Progress("Progress").EndDate("EndDate").Child("SubTasks")).EventMarkers(em
=>
 {
 em.Day("04/10/2019").Label("Project approval and
kick-off").CssClass("e-custom-event-marker").Add();
 }).Render()
```

## ENABLETOOLTIP.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
```

```

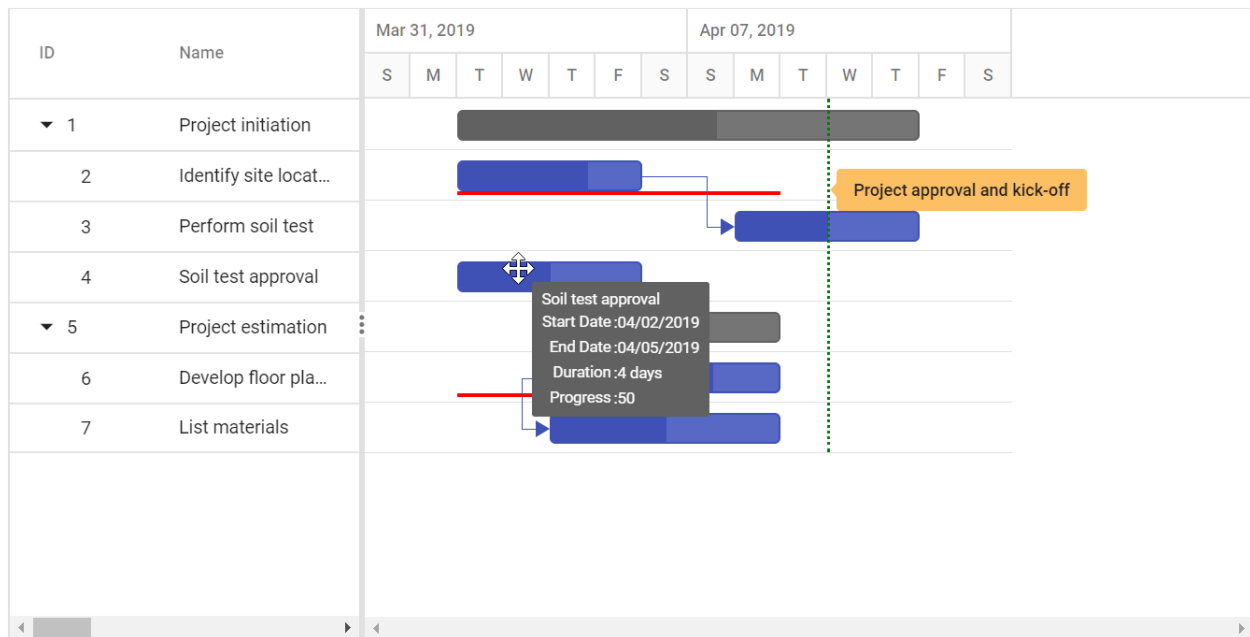
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 BaselineStartDate = new DateTime(2019, 04, 02),
 BaselineEndDate = new DateTime(2019, 04, 08),
 Progress = 70
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Predecessor="2FS",
 Progress = 50
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70,
 BaselineStartDate = new DateTime(2019, 04, 02),
 BaselineEndDate = new DateTime(2019, 04, 06),
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Predecessor = "6SS",
 Progress = 50
 };
 Record2.SubTasks.Add(Child4);

```

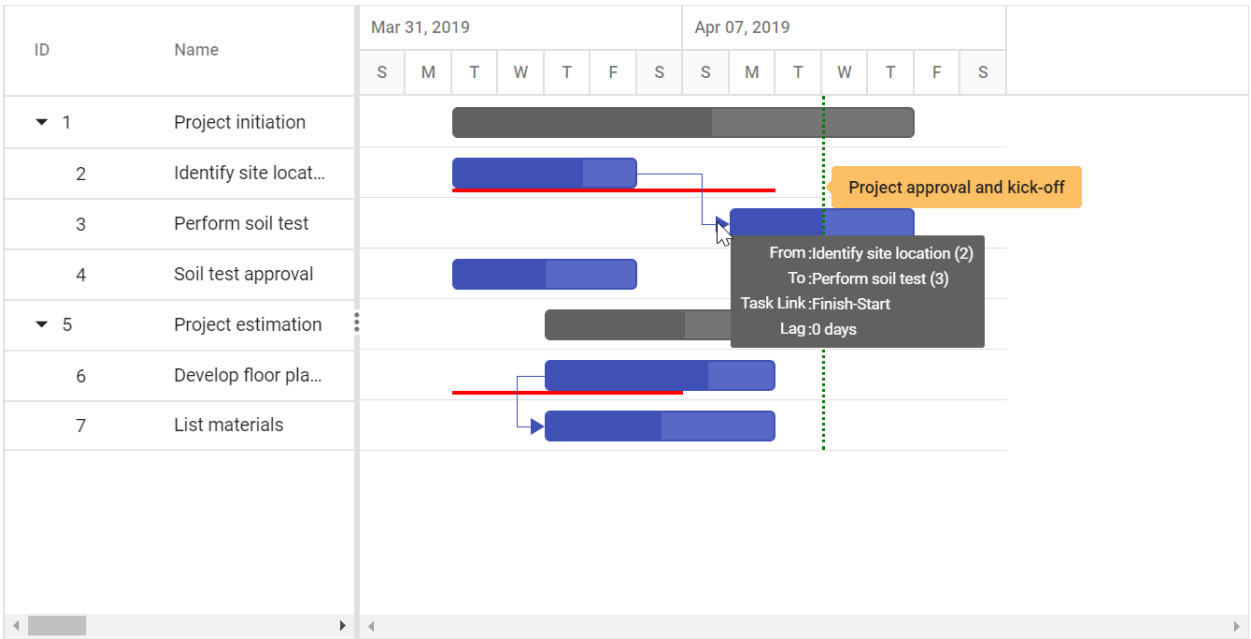
```

Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public DateTime? BaselineStartDate { get; set; }
 public DateTime? BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

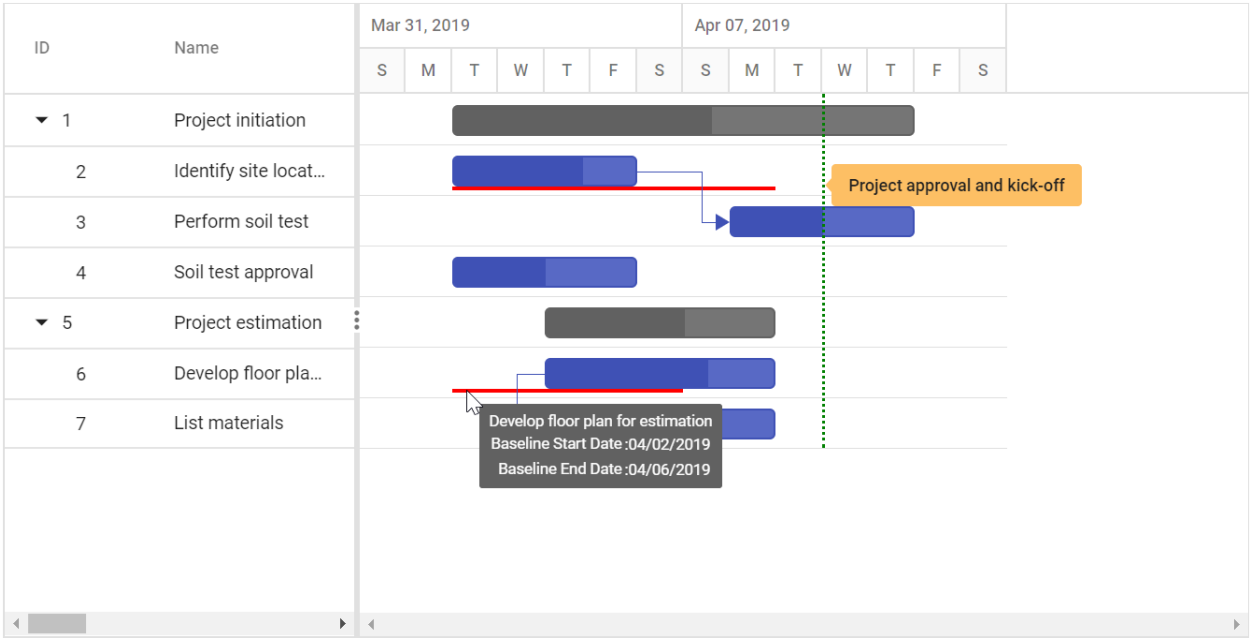
```



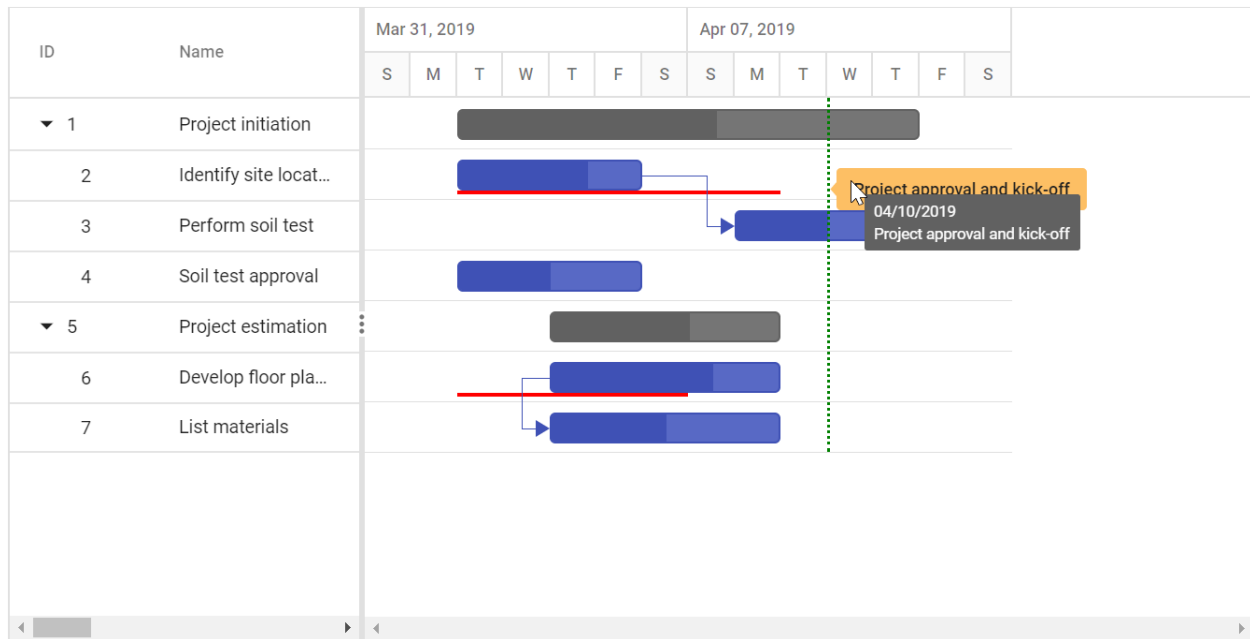
Taskbar Tooltip



Dependency Tooltip



Baseline Tooltip



### Event Marker Tooltip

**Note:** The default value of the [TooltipSettings.ShowTooltip](#) property is true.

### Tooltip template

#### Taskbar tooltip

The default tooltip in the Gantt control can be customized using the [TooltipSettings.Taskbar](#) property. You can map the template script element's ID value or template string directly to this property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Child("SubTasks").Duration("Duration").Progress(
 "Progress")).TooltipSettings(ts =>
ts.Taskbar("#taskbarTooltip")).Render()

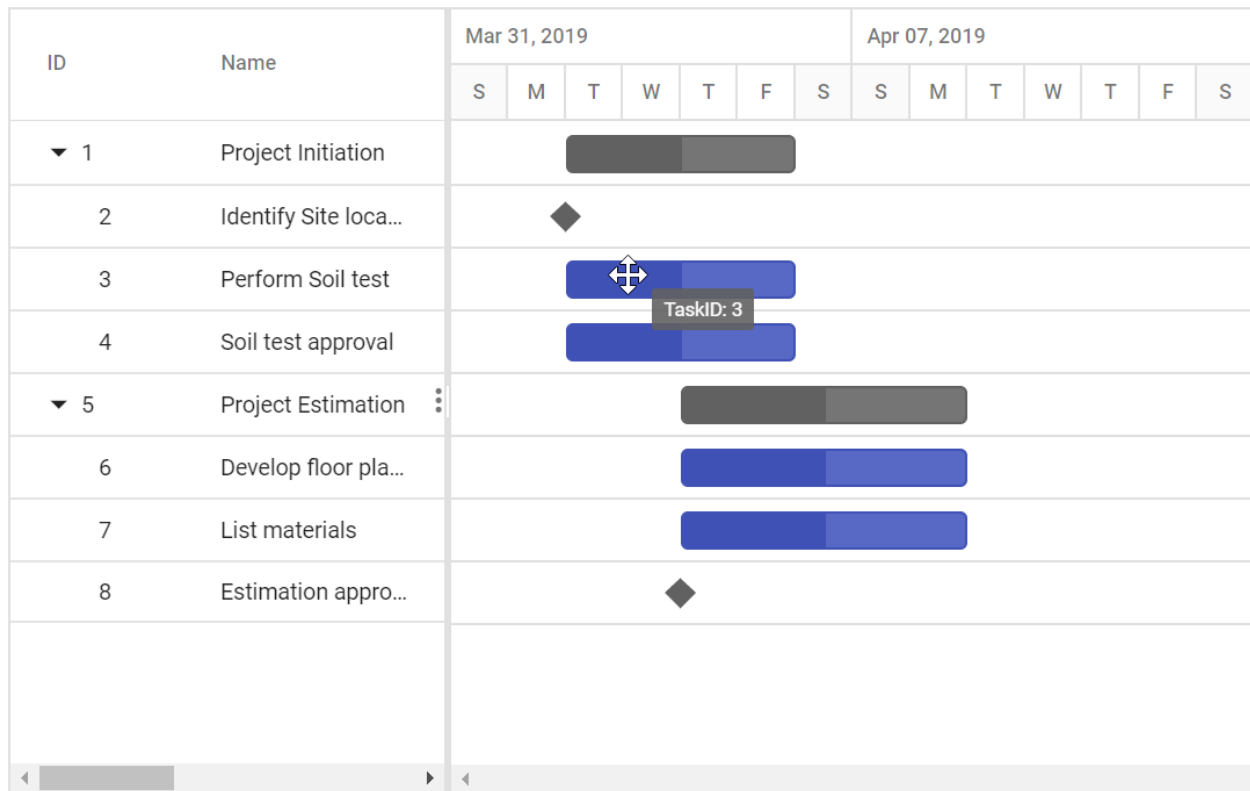
<script type="text/x-jsrender" id="taskbarTooltip">
 <div>TaskID: ${TaskId}</div>
</script>
```

### TASKBARTEMPLATE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

The below screenshot shows the output of above code example.





### Baseline tooltip

A baseline tooltip can be customized using the [TooltipSettings.Baseline](#) property. The following code example shows how to customize the baseline tooltip in Gantt.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").RenderBaseline(true).BaselineColor("red")
 .TaskFields(ts =>
 ts.Id("TaskId").BaselineStartDate("BaselineStartDate").BaselineEndDate("BaselineEndDate").Name("TaskName").StartDate(
 "StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks"))
 .TooltipSettings(ts =>
 ts.Baseline("#baselineTemplate")).Render()

<script type="text/x-jsrender" id="baselineTemplate">
 <div>Baseline StartDate :
 ${this.getFormattedDate(BaselineStartDate)}</div>
</script>
```

### BASELINETEMPLATE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
 public static List<GanttDataSource> ganttData()
 {
```

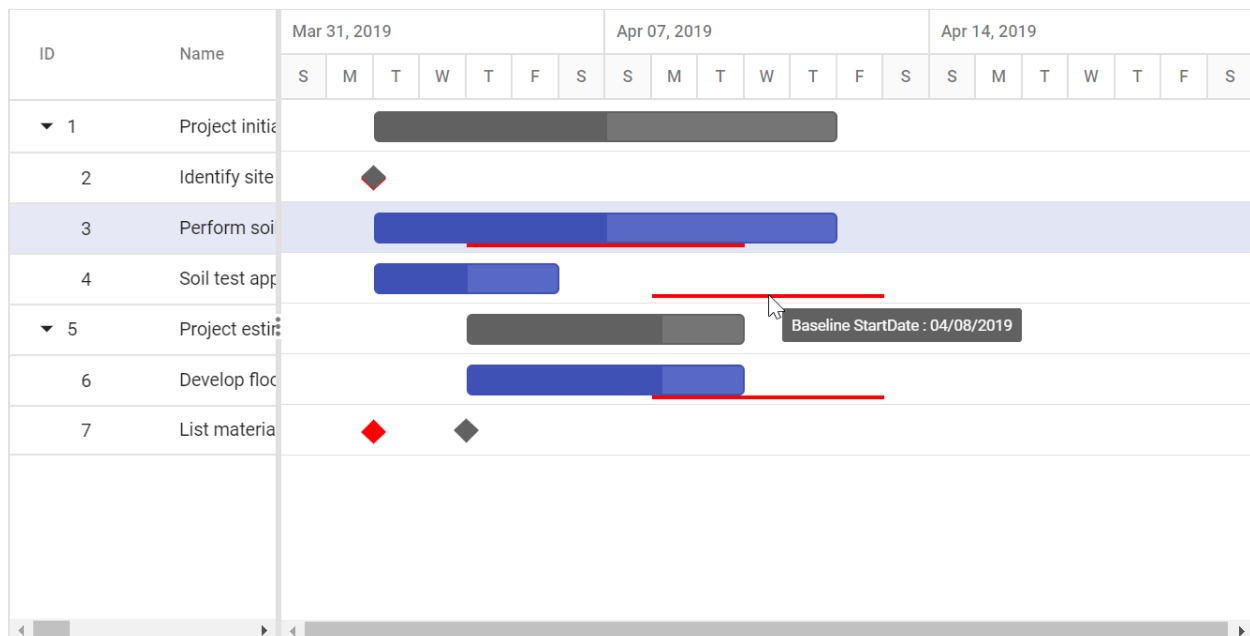
```
List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
GanttDataSource Record1 = new GanttDataSource()
{
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 BaselineStartDate = new DateTime(2019, 04, 02),
 BaselineEndDate = new DateTime(2019, 04, 02),
 Duration = 0,
 Progress = 70,
};
GanttDataSource Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 BaselineStartDate = new DateTime(2019, 04, 04),
 BaselineEndDate = new DateTime(2019, 04, 09),
 Duration = 8,
 Progress = 50
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 BaselineStartDate = new DateTime(2019, 04, 08),
 BaselineEndDate = new DateTime(2019, 04, 12),
 Duration = 4,
 Progress = 50
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 BaselineStartDate = new DateTime(2019, 04, 08),
```

```

 BaselineEndDate = new DateTime(2019, 04, 12),
 Duration = 4,
 Progress = 70
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 BaselineStartDate = new DateTime(2019, 04, 02),
 BaselineEndDate = new DateTime(2019, 04, 02),
 Duration = 0,
 Progress = 50
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public DateTime? BaselineStartDate { get; set; }
 public DateTime? BaselineEndDate { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

The following screenshot shows the template for baseline in Gantt.



### Connector line tooltip

The default connector line tooltip in the Gantt control can be customized using the [TooltipSettings.ConnectorLine](#) property. You can map the value to this property as template script element ID or template string format. The following code example shows how to use the [TooltipSettings.ConnectorLine](#) property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").Duration("Duration")
 .Progress("Progress").Child("SubTasks"
).Dependency("Predecessor")).TooltipSettings(ts =>
ts.ConnectorLine("#dependencyLineTooltip").Render()

<script type="text/x-jsrender" id="dependencyLineTooltip">
 <div>Offset : ${offsetString}</div>
</script>
```

### DEPENDENCYLINETEMPLATE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Predecessor = "2FS"
```

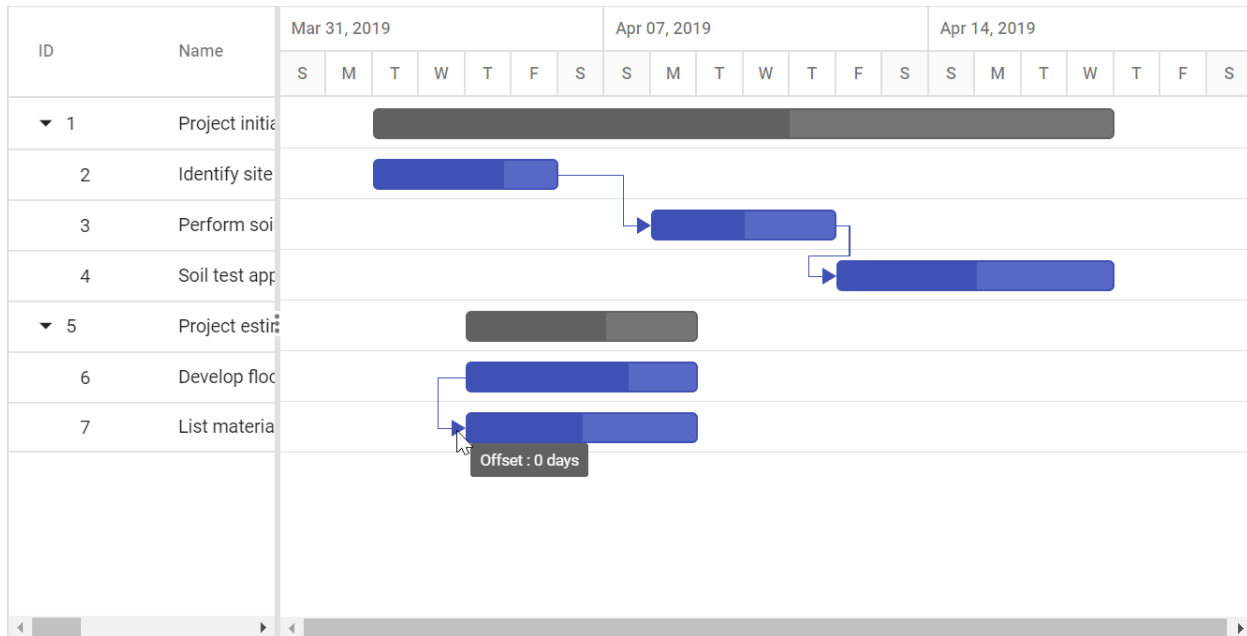
```

 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Predecessor = "3FS"
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Predecessor = "6SS"
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string Predecessor { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

The below screenshot shows the output of above code example.



### Critical Path feature

The critical path in a project is indicated by a single task or a series of tasks. If a task in critical path is delayed, the entire project will be delayed. A task is considered to be critical if any delay to this task would affect the project end date.

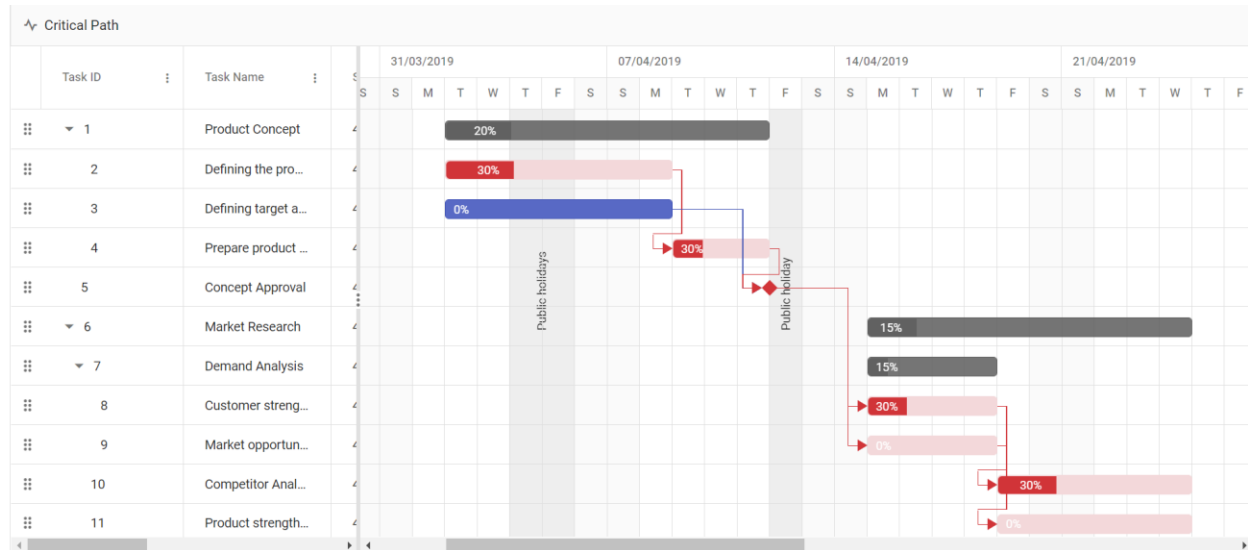
The critical path can be enabled in Gantt by using the built-in toolbar button or `enableCriticalPath` to true.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
).StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
 .Child("SubTasks")
).EnableCriticalPath(true).EditSettings(es =>
 es.AllowAdding(true).AllowEditing(true).AllowDeleting(true)
).Toolbar(new List<string>() { "CriticalPath" }).Render()
```

### CRITICALPATH.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.EditingData();
 return View();
}
```



### Customize taskbar in critical path

The taskbar in critical path can be customized by using `queryTaskbarInfo` event and `isCritical` property of row `data` in the event argument.

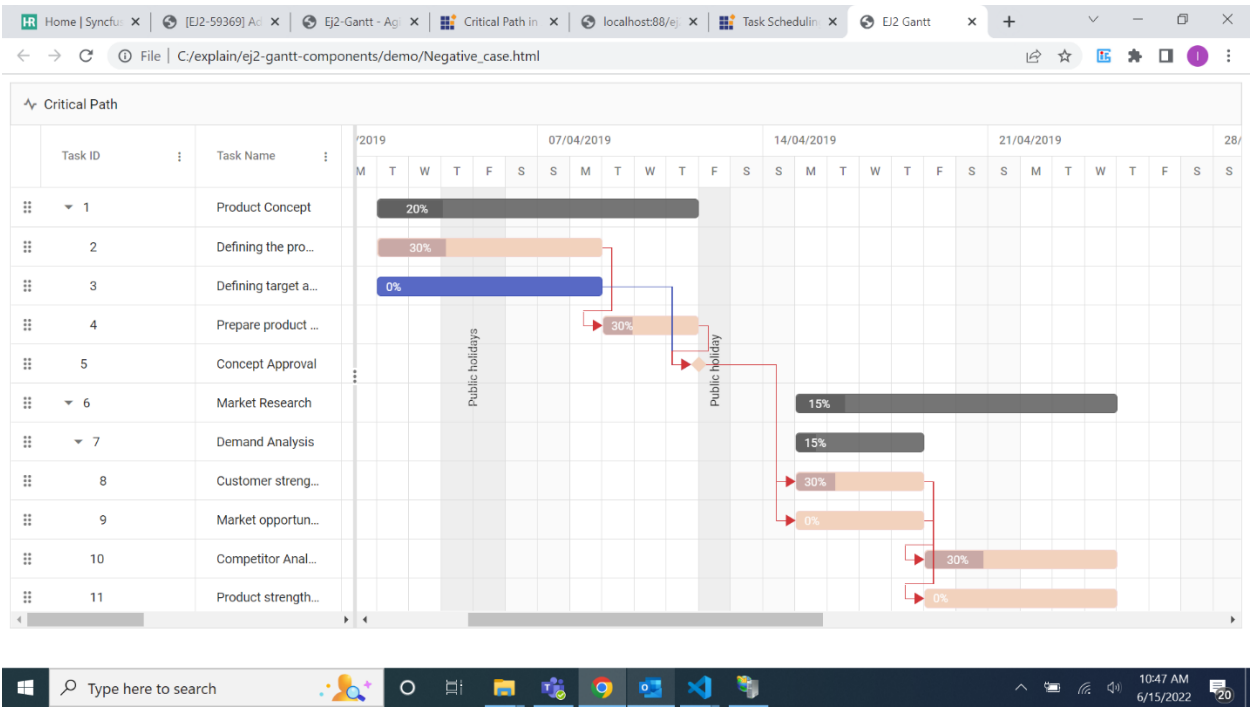
The following code example shows how to customize the critical path taskbar in the Gantt control:

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
 .StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
 .Child("SubTasks")
).EnableCriticalPath(true).EditSettings(es =>
 es.AllowAdding(true).AllowEditing(true).AllowDeleting(true)
).QueryTaskbarInfo("queryTaskbarInfo").Render()
<script>
 function queryTaskbarInfo(args) {
 if ((args.data.isCritical || args.data.slack === '0 day') &&
 !args.data.hasChildRecords) {
 args.taskbarBgColor = 'rgb(242, 210, 189)';
 args.progressBarBgColor = 'rgb(201, 169, 166)';
 }
 }
</script>
```

#### CUSTOMCRITICALPATH.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.EditingData();
 return View();
}
```



Toolbar in Gantt control

The Gantt control provides toolbar support to handle Gantt actions. The [Toolbar](#) property accepts the collection of built-in toolbar items and [ItemModel](#) objects for custom toolbar items.

Built-in toolbar items

Built-in toolbar items execute standard actions of the Gantt control, and these items can be added to toolbar by defining the [Toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
----- -----	
Add	Adds a new record.
Cancel	Cancels the edit state.
CollapseAll	Collapses all the rows.
Delete	Deletes the selected record.
Edit	Edits the selected record.
Indent	Indent the selected record to one level.
Outdent	Outdent the elected record to one level.
ExpandAll	Expands all the rows.
NextTimeSpan	Navigate the Gantt timeline to next time span.
PrevTimeSpan	Navigate the Gantt timeline to previous time span.
Search	Searches the records by the given key.



| Update | Updates the edited record. |

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").Toolbar(new List<string>()
{ "Add", "Cancel", "CollapseAll", "Delete", "Edit", "ExpandAll",
"NextTimeSpan", "PrevTimeSpan", "Search", "Update" }).TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").Child("SubTasks")
).EditSettings(es=>es.AllowEditing(true).AllowAdding(true).AllowDeleting(tru
e)).Render()
```

### DEFAULTITEMS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

**Note:** The [Toolbar](#) has options to define both built-in and custom toolbar items.

#### Custom toolbar items

Custom toolbar items can be added to the toolbar by defining the [Toolbar](#) property as a collection of [ItemModels](#). Actions for this customized toolbar items are defined in the [ToolbarClick](#) event.

By default, the custom toolbar items are at left position. You can change the position by using the [Align](#) property. In the following sample, the **Quick Filter** toolbar item is positioned at right.

### CSHTML

```
@{
 List<object> toolbarItems = new List<object>();
 toolbarItems.Add(new { text = "Quick Filter", tooltipText = "Quick
Filter", id = "toolbarfilter", align = "Right", prefixIcon: "e-quickfilter"
});
}

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.D
ataSource).Height("450px").ToolbarClick(
 "toolbarClick").Toolbar(toolbarItems).TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate"
).EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTas
ks")).Render()

<script>
function toolbarClick(args) {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.filterByColumn("TaskName", "startswith", "Perform");
}
```

```
};
</script>
```

### CUSTOMITEMS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

**Note:** \* The [Toolbar](#) has options to define both built-in and custom toolbar items.

\* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

#### Built-in and custom items in toolbar

The Gantt control has an option to use both built-in and custom toolbar items at the same time.

In the following example, the **ExpandAll** and **CollapseAll** are built-in toolbar items and **Test** is the custom toolbar item.

### CSHTML

```
@{
 List<object> toolbarItems = new List<object>();
 toolbarItems.Add("ExpandAll");
 toolbarItems.Add("CollapseAll");
 toolbarItems.Add(new { text = "Test", tooltipText = "Test", id =
 "Test" });
}

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").ToolbarClick(
 "onToolbarClick").Toolbar(toolbarItems).TaskFields(ts
=> ts.Id("TaskId").Name("TaskName").StartDate("StartDate"
).EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTas
ks")).Render()

<script>
 function onToolbarClick(args) {
 if (args.item.text === 'Test') {
 alert("Custom toolbar click...");
 }
 }
</script>
```

### DEFAULTANDCUSTOMITEMS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Enable/disable toolbar items

You can enable or disable the toolbar items by using the `enableItems` method.

#### CSHTML

```
@{
 List<object> toolbarItems = new List<object>();
 toolbarItems.Add("QuickFilter");
 toolbarItems.Add("ClearFilter");
}

@Html.EJS().Button("enable").Content("Enable").Render()
@Html.EJS().Button("disable").Content("Disable").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").ToolbarClick("onToolbarClick")
).Toolbar(toolbarItems).TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration")
).Progress("Progress").Child("SubTasks")).Render()

<script>

 document.getElementById("enable").addEventListener('click', () => {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.toolbarModule.enableItems([ganttObj.element.id +
'_QuickFilter', ganttObj.element.id + '_ClearFilter'], true); // enable
toolbar items.
 });
 document.getElementById("disable").addEventListener('click', () => {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.toolbarModule.enableItems([ganttObj.element.id +
'_QuickFilter', ganttObj.element.id + '_ClearFilter'], false); // disable
toolbar items.
 });
 function onToolbarClick(args) {
 if (args.item.text === 'QuickFilter') {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.filterByColumn("TaskName", "startswith",
"Perform");
 }
 if (args.item.text === 'ClearFilter') {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.clearFiltering();
 }
 }
}

</script>
```

#### ENABLE-DISABLETOOLBARITEMS.CS

```
public IActionResult Index()
```

```
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Add input elements to toolbar

In the Gantt toolbar, you can add EJ2 editor elements like a numeric text box, a drop-down list, and date picker controls. The following code snippets demonstrate how to add EJ2 editors to the Gantt toolbar:

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").Toolbar(new List<string>()
{
 "Add", "Cancel", "CollapseAll", "Delete", "Edit", "ExpandAll",
 "NextTimeSpan", "PrevTimeSpan", "Search", "Update", {type:'Input', template:
 new NumericTextBox({ format: 'c2', value:1, width:150 })} })
.TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")

).EditSettings(es=>es.AllowEditing(true).AllowAdding(true).AllowDeleting(true)).Render()
```

#### INPUT-ELEMENTS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Managing Tasks in ASP.NET MVC Gantt Component

The Gantt component has options to dynamically insert, delete, and update tasks in the project. The primary key column is necessary to manage the tasks and perform CRUD operations in Gantt. To define the primary key, set the [Columns.IsPrimaryKey](#) property to `true` in the particular column.

The following code example shows you how to enable the cell editing in Gantt control.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(

"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")

).EditSettings(es=>es.AllowEditing(true).Mode(Syncfusion.EJ2.Gantt.EditMode.Auto)).Render()
```

#### ENABLECELLEDITING.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Cell edit type and its params

The [columns.editType](#) is used to define the edit type for any particular column. You can set the [columns.editType](#) based on data type of the column.

- numeric edit - [NumericTextBox](#) component for integers, double, and decimal data types.
- default edit - [TextBox](#) component for string data type.
- dropdown edit - [DropDownList](#) component to show all unique values related to that field.
- boolean edit - [CheckBox](#) component for boolean data type.
- date picker edit - [DatePicker](#) component for date data type.
- date time picker edit - [DateTimePicker](#) component for date time data type.

Also, you can customize the behavior of the editor component through the [columns.edit.params](#).

The following table describes cell edit type component and their corresponding edit params of the column.

Edit Type	Component	Example
numericedit	<a href="#">NumericTextBox</a>	params: { decimals: 2, value: 5 }
dropdownedit	<a href="#">DropDownList</a>	params: { value: 'Germany' }
booleanedit	<a href="#">Checkbox</a>	params: { checked: true }
datepickeredit	<a href="#">DatePicker</a>	params: { format: 'dd.MM.yyyy' }
datetimepickeredit	<a href="#">DateTimePicker</a>	params: { value: new Date() }

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Columns(col =>
 {
 col.Field("TaskId").Add();
 col.Field("TaskName").HeaderText("Task Name").Add();
 col.Field("StartDate").Add();
 col.Field("Duration").EditType("numericedit").Edit(new {
@params = new { min = 1 } }).ValueAccessor("durationFormat").Add();
 col.Field("Progress").EditType("numericedit").Edit(new {
@params = new Syncfusion.EJ2.Inputs.NumericTextBox() { ShowSpinButton=false }
 }).Add();
 }) .EditSettings(es=>es.AllowEditing(true)).Render()

<script>
```

```
function durationFormat(field, data, column) {
 return data[field];
}
</script>
```

### EDITPARAMS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/editParams.png)

### Cell edit template

The cell edit template is used to create a custom component for a particular column by invoking the following functions:

- **create** - It is used to create the element at the time of initialization.
- **write** - It is used to create the custom component or assign default value at the time of editing.
- **read** - It is used to read the value from the component at the time of save.
- **destroy** - It is used to destroy the component.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Columns(col =>
{
 col.Field("TaskId").Add();
 col.Field("TaskName").HeaderText("Task Name").Edit(new {
create = "create", read = "read", destroy = "destroy", write = "write"
}).Add();

 col.Field("StartDate").Add();
 col.Field("Duration").Add();
 col.Field("Progress").Add();
}).EditSettings(es=>es.AllowEditing(true)).Render()

<script>
var elem;
var dropdownlistObj;
function create(args) {
 elem = document.createElement('input');
 return elem;
}
function write(args) {
 var gantt = document.getElementById("Gantt").ej2_instances[0];
 dropdownlistObj = new ej.dropdowns.DropDownList({
 dataSource: gantt.treeGrid.grid.dataSource,
 fields: { value: 'TaskName' },
 value: args.rowData[args.column.field],
```

```

 floatLabelType: 'Auto',
 });
 dropdownlistObj.appendTo(elem);
}
function destroy() {
 dropdownlistObj.destroy();
}
function read(args) {
 return dropdownlistObj.value;
}
</script>

```

### EDITTEMPLATE.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

![[Alt text]](images/editTemplate.png)

Disable editing for particular column

You can disable editing for particular columns, by using the [columns.allowEditing](#) property.

In the following demo, editing is disabled for the **TaskName** column.

### CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Columns(col =>
{
 col.Field("TaskId").Add();
 col.Field("TaskName").HeaderText("TaskName").AllowEditing(false).Add();
 col.Field("StartDate").Add();
 col.Field("Duration").Add();
 col.Field("Progress").Add();
}).EditSettings(es=>es.AllowEditing(true)).Render()

```

### DISABLEEDITING.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

### Read-only gantt

In Gantt, all create, update, delete operations can be disabled by setting `readOnly` property as `true`. The following sample demonstrates, render Gantt chart as read only.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").Toolbar(new List<string>()
{ "Add", "Cancel", "CollapseAll", "Delete", "Edit", "ExpandAll",
"NextTimeSpan", "PrevTimeSpan", "Search", "Update" }).TaskFields(ts =>
ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).EnableContextMenu(true).AllowSorting(true).AllowResizing(true).ReadOnly(true).EditSettings(es=>
es.AllowEditing(true).AllowAdding(true).AllowDeleting(true).AllowTaskbarEditing(true)).Render()
```

#### READONLY.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

![Alt text](images/readOnly.PNG)

![Alt text](images/readOnlyEdit.png)

### Open new task dialog with default values

You can set default values when new task dialog opens using `actionBegin` event when `requestType` is `beforeOpenAddDialog`.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts => ts.Id("TaskId")
).Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).EditSettings(es=>
es.AllowAdding(true)).Toolbar(new List<string>() { "Add"
}).ActionBegin("onActionBegin").Render()
<script>
function onActionBegin(args) {
 if (args.requestType == 'beforeOpenAddDialog') {
 args.rowData.TaskName = 'Gantt';
 args.rowData.Progress = 70;
 args.rowData.ganttProperties.taskName = 'Gantt';
 args.rowData.ganttProperties.progress = 70;
 }
}
</script>
```



**TASK-DIALOG-DEFAULT-VALUE.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 ViewBag.projectResources = projectResources();
 return View();
}

public static List<GanttResources> projectResources()
{
 List<GanttResources> GanttResourcesCollection = new
List<GanttResources>();
 GanttResources Record1 = new GanttResources()
 {
 ResourceId = 1,
 ResourceName = "Martin Tamer"
 };
 GanttResources Record2 = new GanttResources()
 {
 ResourceId = 2,
 ResourceName = "Rose Fuller"
 };
 GanttResources Record3 = new GanttResources()
 {
 ResourceId = 3,
 ResourceName = "Margaret Buchanan"
 };
 GanttResources Record4 = new GanttResources()
 {
 ResourceId = 4,
 ResourceName = "Fuller King"
 };
 GanttResources Record5 = new GanttResources()
 {
 ResourceId = 5,
 ResourceName = "Davolio Fuller"
 };
 GanttResources Record6 = new GanttResources()
 {
 ResourceId = 6,
 ResourceName = "Van Jack"
 };
 GanttResources Record7 = new GanttResources()
 {
 ResourceId = 7,
 ResourceName = "Fuller Buchanan"
 };
 GanttResources Record8 = new GanttResources()
 {
 ResourceId = 8,
 ResourceName = "Jack Davolio"
 };
 GanttResources Record9 = new GanttResources()
 {
 ResourceId = 9,
 ResourceName = "Tamer Vinet"
 };
};
```

```

GanttResources Record10 = new GanttResources()
{
 ResourceId = 10,
 ResourceName = "Vinnet Fuller"
};
GanttResources Record11 = new GanttResources()
{
 ResourceId = 11,
 ResourceName = "Bergs Anton"
};
GanttResources Record12 = new GanttResources()
{
 ResourceId = 12,
 ResourceName = "Construction Supervisor"
};
GanttResourcesCollection.Add(Record1);
GanttResourcesCollection.Add(Record2);
GanttResourcesCollection.Add(Record3);
GanttResourcesCollection.Add(Record4);
GanttResourcesCollection.Add(Record5);
GanttResourcesCollection.Add(Record6);
GanttResourcesCollection.Add(Record7);
GanttResourcesCollection.Add(Record8);
GanttResourcesCollection.Add(Record9);
GanttResourcesCollection.Add(Record10);
GanttResourcesCollection.Add(Record11);
GanttResourcesCollection.Add(Record12);
return GanttResourcesCollection;
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 isParent= true
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 1 },
 info="Measure the total property area allotted for
construction"
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",

```

```

 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 2, 3, 5 },
 info= "Obtain an engineered soil test of lot where
construction is planned."+
 "From an engineer or company specializing in soil
testing"
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Dependency = "3FS",
 Progress = 50
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 isParent= true
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ResourceId = new int[] { 4 },
 info = "Develop floor plans and obtain a materials list for
estimations"
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 Dependency="6SS",
 ResourceId = new int[] { 4, 8 },
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

```

```

public class GanttResources
{
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string Dependency { get; set; }
 public bool isParent { get; set; }
 public string info { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
}

```

![Gantt new task dialog with default values](images/task-dialog-with-default-values.png)

Troubleshoot: Editing works only when primary key column is defined

Editing feature requires a primary key column for CRUD operations. While defining columns in Gantt using the [columns](#) property, it is mandatory that any one of the columns, must be a primary column. By default, the [id](#) column will be the primary key column. If [id](#) column is not defined, we need to enable [isPrimaryKey](#) for any one of the columns defined in the [columns](#) property.

Touch interaction

The Gantt control editing actions can be achieved using the double tap and tap and drag actions on a element.

The following table describes different types of editing modes available in Gantt.

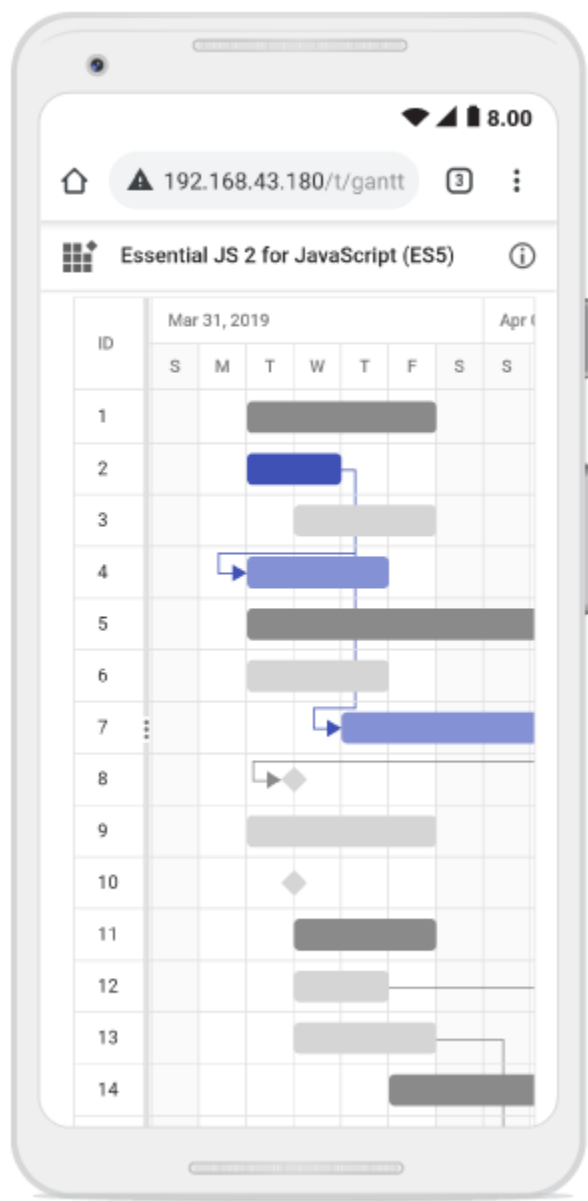
Action	Description
----- -----	
<a href="#">Cell editing</a>	To perform <b>double tap</b> on a specific cell, initiate the cell to be in edit state.
<a href="#">Dialog editing</a>	To perform <b>double tap</b> on a specific row, initiate the edit dialog to be opened.
<a href="#">Taskbar editing</a>	Taskbar editing action is initiated using the <b>tap</b> action on the taskbar.   <b>Parent taskbar</b> : Once you tap on the parent taskbar, it will be changed to editing state. Perform only dragging
	action on parent taskbar editing.  
	: Once you tap the child taskbar, it will be changed to editing state.  
	 <b>Dragging taskbar</b> : To drag a taskbar to the
	left or right in editing state.     <b>Resizing taskbar</b> : To resize a taskbar, drag the left/right resize
	icon.     <b>Progress resizing</b> : To change the progress, drag the progress resize icon to the left or
	right direction.



Task dependency editing

You can tap the left/right connector point to initiate [task dependencies](#) edit mode and again tap another taskbar to establish the dependency line between two taskbars.

The following table explains the taskbar state in dependency edit mode.



Taskbar state	Description
----- -----	
Parent taskbar	You cannot create dependency relationship to parent tasks.  



|

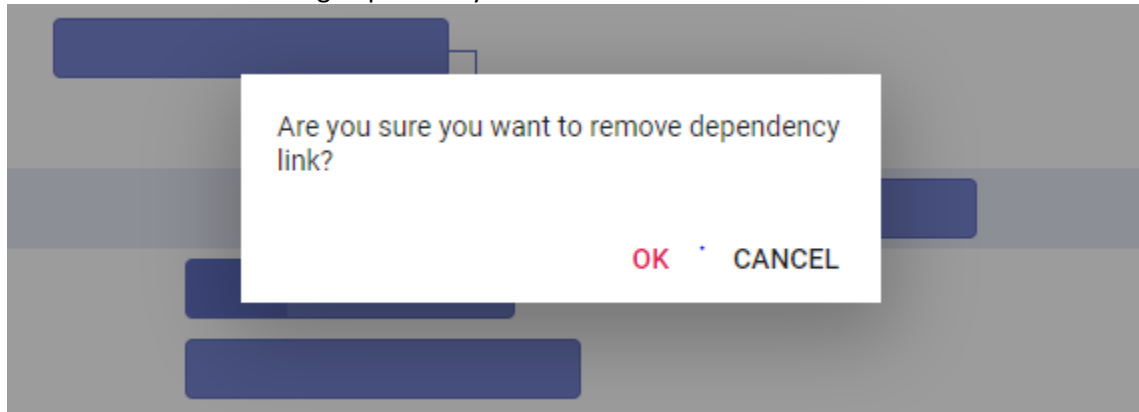
**|Taskbar without dependency |** If you tap a valid child taskbar, it will create FS type dependency line between tasks, otherwise exits from task dependency edit mode. <br>



**|Taskbar with dependency |** If you tap the second taskbar, which has already been directly connected, it will ask to remove it. <br>



**|Removing dependency |** Once you tap the taskbar with direct dependency, then confirmation dialog will be shown for removing dependency. <br>



**Note:** In mobile device, you cannot create dependency other than FS by taskbar editing. By using cell/dialog editing, you can add all type of dependencies.

### Taskbar editing tooltip

The taskbar editing tooltip can be customized using the [TooltipSettings.Editing](#) property. The following code example shows how to customize the taskbar editing tooltip in Gantt.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate")
).EndDate("EndDate").Child("SubTasks").Duration("Duration").Progress("Progress")
).TooltipSettings(ts => ts.Editing("#editingTooltip")
).EditSettings(es=>es.AllowTaskbarEditing(true)).Render()

<script type="text/x-jsrender" id="editingTooltip">
 <div>Duration : ${duration}</div>
</script>
```

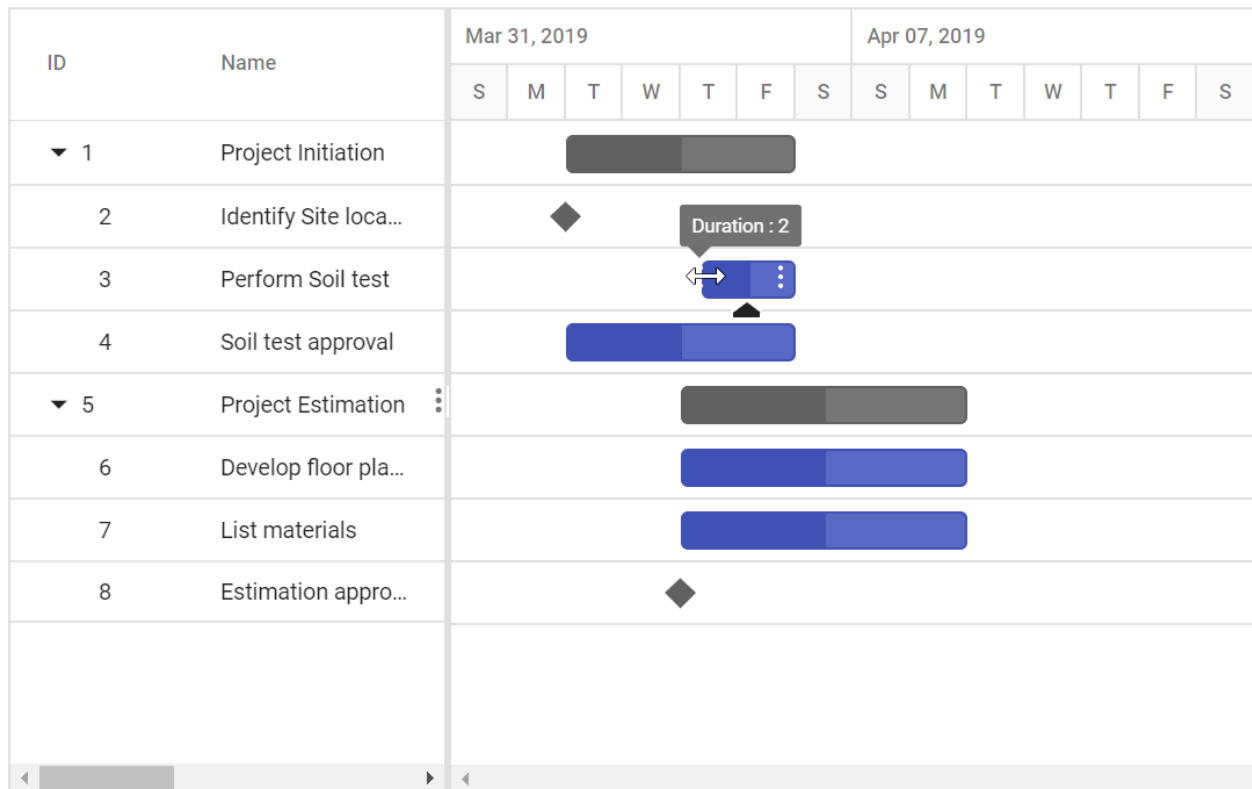
### EDITINGTEMPLATE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

```
}

```

The below screenshot shows the output of above code example.



### Scrolling in Gantt Control

The scrollbar will be displayed in the gantt when content exceeds the element [width](#) or [height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the gantt exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the grid pane size.
- The [height](#) and [width](#) are used to set the gantt height and width, respectively.

**Note:** The default value for [height](#) and [width](#) is auto.

### Set width and height

To specify the [width](#) and [height](#) of the scroller in the pixel, set the pixel value to a number.

The following code example shows how to set height and width in the Gantt control:

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").Width("600px").TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName")

```

```

).StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true)
).Render()

```

### **WIDTH-HEIGHT.CS**

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.EditingData();
 return View();
}

```

Responsive with the parent container

Specify the [width](#) and [height](#) as **100%** to make the gantt element fill its parent container.

Setting the [height](#) to **100%** requires the gantt parent element to have explicit height.

The following code example shows how to set height and width in the Gantt control:

### **CSHTML**

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("100%").Width("100%").TaskFields(ts =>
ts.Id("TaskId").Name("TaskName")

).StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true)
).Render()

```

### **RESPONSIVE.CS**

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.EditingData();
 return View();
}

```

Scroll To Date method

In the Gantt control, When We use the [scrollToDate](#) method, it will scroll the timeline horizontally to the date that we specified in the method's argument.

The following code examples show how the scroll To Date method works in Gantt:

### **CSHTML**

```

@Html.EJS().Button("scrollToDate").Content("Change Scroll Position").CssClass("e-primary").Render()

```



```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Render()

<script>
 document.getElementById('scrollToDate').addEventListener('click',
function (args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.scrollToDate('05/20/2019');
});
</script>
```

### **SCROLLTODATE.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Set the vertical scroll position

In the Gantt control, you can set the vertical scroller position dynamically by clicking the custom button using the `setScrollTop` method.

### **CSHTML**

```
@Html.EJS().Button("setScrollTop").Content("Change Scroll
Position").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).Render()

<script>
 document.getElementById('setScrollTop').addEventListener('click',
function (args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.ganttChartModule.scrollObject.setScrollTop(300);
});
</script>
```

### **SETSCROLLTOP.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
}
```

```
return View();
}
```

## Virtual Scrolling in Gantt

Virtual Scroll support in Gantt allows you to load large amount of data without performance degradation. To enable Virtual Scrolling, you need to inject `VirtualScroll` module in Gantt.

### Row virtualization

Row virtualization allows you to load and render a large number of tasks in Gantt with effective performance. In this mode, all tasks are fetched initially from the datasource and rendered in the DOM within a compact viewport area.

The number of records displayed in the Gantt is determined by the height.

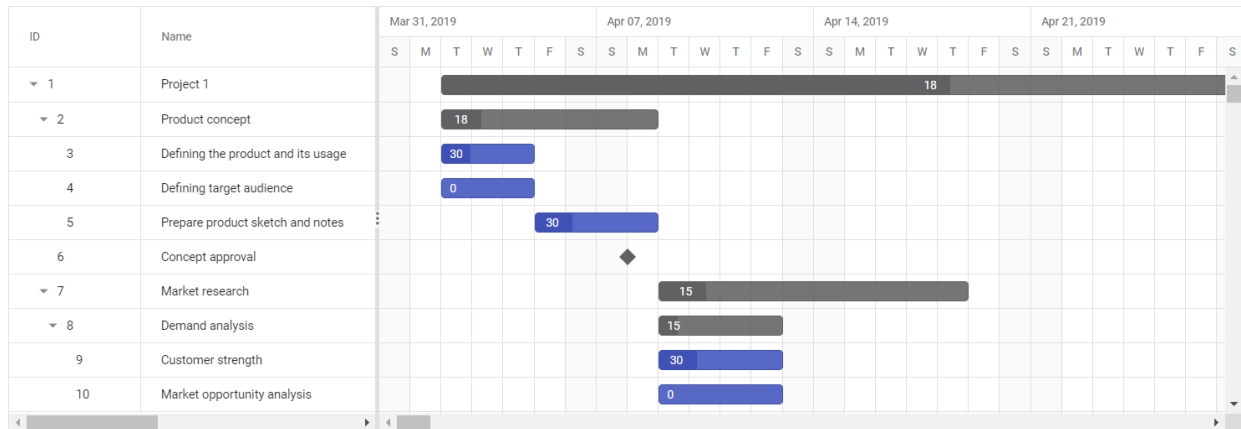
This mode can be enable by setting the `EnableVirtualization` property to `true`.

### CSHTML

```
@(Html.EJS().Gantt("VirtualScroll").DataSource((IEnumerable<object>)ViewBag.
DataSource).Height("450px").EnableVirtualization(true).HighlightWeekends(true)
.AllowSelection(true)
 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress")
 .Child("SubTasks").ParentID("ParentID"))
 .LabelSettings(ls=>ls.TaskLabel("Progress")).Columns(col =>
 {
 col.Field("TaskId").Width(140).Add();
 col.Field("TaskName").Width(250).Add();
 col.Field("StartDate").Add();
 col.Field("EndDate").Add();
 col.Field("Duration").Add();
 col.Field("Progress").Add();
 }).SplitterSettings(ss => ss.ColumnIndex(2))
 .Render()
)
```

### VIRTUAL-SCROLL.CS

```
public IActionResult Index()
{
 ViewBag.dataSource = GanttData.VirtualData();
 return View();
}
```



### Timeline virtualization

Timeline virtualization allows you to load a data source having large timespan with high performance. Initially, it renders the timeline with thrice the width of the gantt element, while other timeline cells render on-demand during horizontal scrolling.

This mode can be enable by setting the [EnableTimelineVirtualization](#) property to `true`.

### CSHTML

```
@(Html.EJS().Gantt("VirtualScroll").DataSource((IEnumerable<object>)ViewBag.
DataSource).Height("450px").EnableVirtualization(true).EnableTimelineVirtual
ization(true).HighlightWeekends(true).AllowSelection(true)
.TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress")
.Child("SubTasks").ParentID("ParentID"))
.LabelSettings(ls=>ls.TaskLabel("Progress")).EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true).AllowTaskbarEdit
ing(true).ShowDeleteConfirmDialog(true))
.AutoCalculateDateScheduling(false).Toolbar(new List<string>()
{ "Add", "Edit", "Update", "Delete", "Cancel", "ExpandAll", "CollapseAll",
"Indent", "Outdent" }).Columns(col =>
{
col.Field("TaskId").Width(140).Add();
col.Field("TaskName").Width(250).Add();
col.Field("StartDate").Add();
col.Field("EndDate").Add();
col.Field("Duration").Add();
col.Field("Progress").Add();
}).SplitterSettings(ss => ss.ColumnIndex(2))
.Render()
)
```

### TIMELINE-VIRTUAL-SCROLL.CS

```
public IActionResult Index()
{
 ViewBag.dataSource = GanttData.VirtualData();
 return View();
}
```

### Limitations for virtual scroll

- Due to the element height limitation in browsers, the maximum number of records loaded is limited by the browser capacity.
- Cell selection will not be persisted.
- The number of records rendered will be determined by the `Height` property.
- It is necessary to mention the height of the Gantt in pixels when enabling Virtual Scrolling.

### Resources in Syncfusion ASP.NET MVC Gantt Component

In Gantt, the resources are represented by staff, equipment and materials etc. In Gantt control you can show or allocate the resources (human resources) for each task.

#### Resource collection

The resource collection contains details about resources that are used in the project. Resources are JSON object that contains id, name, unit and group of the resources and this collection is mapped to the Gantt control using the [resources](#) property. These resource fields are mapped to the Gantt control using the [resourceFields](#) property.

|Resource fields | Description|

|----|----|

|[id](#) | This field is used to assign resources to the tasks.|

|[name](#) | This field is used to map the resource names. These names are displayed as one of Gantt columns and also can display as labels using the [labelSettings](#) property.|

|[unit](#) | It indicates the amount of work that can be done by a resource for the task in a day.|

|[group](#) | This field is used to group the resources and the tasks assigned to that particular resource into category.|

The following code snippets shows resource collection and how it assigned to Gantt control.

#### CSHTML

```
@model List<GanttSample.Controllers.GanttDataSource>
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)Model).AllowFiltering(true).AllowSorting(true).Height("450px").TaskFields(ts =>
ts.Id("TaskId")

).Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks").ResourceInfo(
 "ResourceId").LabelSettings(ls =>
ls.RightLabel("${if(ResourceId)} ${ResourceId} ${/if}")
).ResourceFields(rf =>
rf.Id("ResourceId").Name("ResourceName")).Resources((IEnumerable<object>)ViewBag.projectResources).Render()
```

#### ASSIGNRESOURCE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
}
```

```
ViewBag.projectResources = projectResources();
return View();
}
public static List<GanttResources> projectResources()
{
 List<GanttResources> GanttResourcesCollection = new
List<GanttResources>();
 GanttResources Record1 = new GanttResources()
 {
 ResourceId = 1,
 ResourceName = "Martin Tamer"
 };
 GanttResources Record2 = new GanttResources()
 {
 ResourceId = 2,
 ResourceName = "Rose Fuller"
 };
 GanttResources Record3 = new GanttResources()
 {
 ResourceId = 3,
 ResourceName = "Margaret Buchanan"
 };
 GanttResources Record4 = new GanttResources()
 {
 ResourceId = 4,
 ResourceName = "Fuller King"
 };
 GanttResources Record5 = new GanttResources()
 {
 ResourceId = 5,
 ResourceName = "Davolio Fuller"
 };
 GanttResources Record6 = new GanttResources()
 {
 ResourceId = 6,
 ResourceName = "Van Jack"
 };
 GanttResources Record7 = new GanttResources()
 {
 ResourceId = 7,
 ResourceName = "Fuller Buchanan"
 };
 GanttResources Record8 = new GanttResources()
 {
 ResourceId = 8,
 ResourceName = "Jack Davolio"
 };
 GanttResources Record9 = new GanttResources()
 {
 ResourceId = 9,
 ResourceName = "Tamer Vinet"
 };
 GanttResources Record10 = new GanttResources()
 {
 ResourceId = 10,
 ResourceName = "Vinet Fuller"
 };
};
```

```
GanttResources Record11 = new GanttResources()
{
 ResourceId = 11,
 ResourceName = "Bergs Anton"
};
GanttResources Record12 = new GanttResources()
{
 ResourceId = 12,
 ResourceName = "Construction Supervisor"
};
GanttResourcesCollection.Add(Record1);
GanttResourcesCollection.Add(Record2);
GanttResourcesCollection.Add(Record3);
GanttResourcesCollection.Add(Record4);
GanttResourcesCollection.Add(Record5);
GanttResourcesCollection.Add(Record6);
GanttResourcesCollection.Add(Record7);
GanttResourcesCollection.Add(Record8);
GanttResourcesCollection.Add(Record9);
GanttResourcesCollection.Add(Record10);
GanttResourcesCollection.Add(Record11);
GanttResourcesCollection.Add(Record12);
return GanttResourcesCollection;
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 1 },
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 2, 3, 5 },
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
```

```

 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ResourceId = new int[] { 4 },
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ResourceId = new int[] { 4, 8 },
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttResources
{
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
}

```

```
}
```

### Assign resource

We can assign resources for a task at initial load, using the resource id value of the resources as a collection. This collection is mapped from the dataSource to the Gantt control using the [resourceInfo](#) property.

Resources are assigned to tasks in following ways.

#### Assign resource alone

If the unit is not specified for specific resource, the amount of work done will be consider as 100% by default. In such cases, the resource unit will not be displayed in Gantt UI.

```
`html
```

```
GanttDataSource Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 1 },
};
`
```

#### Assign resources with unit

We can assign the quantity of work done by the resources for the specific task as like below code snippet.

```
`html
```

```
GanttDataSource Record1Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify Site location",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 2,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel>
{

```



```
new ResourceModel{ ResourceId = 1, Unit = 50 }
}
};
`
```

When resource unit is defined in resource collection, the amount of work done by that particular resource will be same for all the tasks.

The following code snippet shows how to assign the resource for each task and map to Gantt control.

### CSHTML

```
@ (Html.EJS().Gantt("ResourceAllocation").DataSource((IEnumerable<object>
) ViewBag.DataSource)
.TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress")
.ResourceInfo("Resources").Child("SubTasks"))
.ResourceFields(rf =>
rf.Id("ResourceId").Name("ResourceName").Unit("ResourceUnit"))
.Resources((IEnumerable<object>) ViewBag.projectResources)
.Columns(col =>
{
col.Field("TaskId").Visible(false).Add();
col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
col.Field("Resources").HeaderText("Resources").Width(160).Add();
col.Field("Duration").Width(100).Add();
})
.LabelSettings(ls => ls.RightLabel("Resources"))
.Render()
)
```

### RESOURCEUNIT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication8Core.Models;
using Syncfusion.EJ2.Gantt;
namespace WebApplication8Core.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.DataSource = ganttData();
 ViewBag.projectResources = projectResources();
 return View();
 }
 public static List<GanttResources> projectResources()
 {

```

```
List<GanttResources> GanttResourcesCollection = new
List<GanttResources>();
GanttResources Record1 = new GanttResources()
{
 ResourceId = 1,
 ResourceName = "Martin Tamer"
};
GanttResources Record2 = new GanttResources()
{
 ResourceId = 2,
 ResourceName = "Rose Fuller"
};
GanttResources Record3 = new GanttResources()
{
 ResourceId = 3,
 ResourceName = "Margaret Buchanan"
};
GanttResources Record4 = new GanttResources()
{
 ResourceId = 4,
 ResourceName = "Fuller King"
};
GanttResources Record5 = new GanttResources()
{
 ResourceId = 5,
 ResourceName = "Davolio Fuller"
};
GanttResources Record6 = new GanttResources()
{
 ResourceId = 6,
 ResourceName = "Van Jack"
};
GanttResources Record7 = new GanttResources()
{
 ResourceId = 7,
 ResourceName = "Fuller Buchanan"
};
GanttResources Record8 = new GanttResources()
{
 ResourceId = 8,
 ResourceName = "Jack Davolio"
};
GanttResources Record9 = new GanttResources()
{
 ResourceId = 9,
 ResourceName = "Tamer Vinet"
};
GanttResources Record10 = new GanttResources()
{
 ResourceId = 10,
 ResourceName = "Vinet Fuller"
};
GanttResources Record11 = new GanttResources()
{
 ResourceId = 11,
 ResourceName = "Bergs Anton"
};
```

```
GanttResources Record12 = new GanttResources()
{
 ResourceId = 12,
 ResourceName = "Construction Supervisor"
};
GanttResourcesCollection.Add(Record1);
GanttResourcesCollection.Add(Record2);
GanttResourcesCollection.Add(Record3);
GanttResourcesCollection.Add(Record4);
GanttResourcesCollection.Add(Record5);
GanttResourcesCollection.Add(Record6);
GanttResourcesCollection.Add(Record7);
GanttResourcesCollection.Add(Record8);
GanttResourcesCollection.Add(Record9);
GanttResourcesCollection.Add(Record10);
GanttResourcesCollection.Add(Record11);
GanttResourcesCollection.Add(Record12);
return GanttResourcesCollection;
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify Site location",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 2,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 1, Unit = 50 }
 }
 };
 GanttDataSource Record1Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 4,
 Progress = 30,
 Work = 20,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 2, Unit = 70 }
 }
 };
};
```

```
GanttDataSource Record1Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 1,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel> {
 new ResourceModel { ResourceId = 3, Unit = 25 },
 new ResourceModel { ResourceId = 1, Unit = 75 },
 }
};
Record1.SubTasks.Add(Record1Child1);
Record1.SubTasks.Add(Record1Child2);
Record1.SubTasks.Add(Record1Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record2Child1 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 3,
 Progress = 30,
 Work = 30,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 4, Unit = 50 },
 new ResourceModel{ ResourceId = 2, Unit = 70 }
 }
};
GanttDataSource Record2Child2 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 3,
 Progress = 30,
 Work = 40,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 6, Unit = 40 }
 }
};
GanttDataSource Record2Child3 = new GanttDataSource()
{
 TaskId = 8,
 TaskName = "Estimation approval",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 2,
```

```

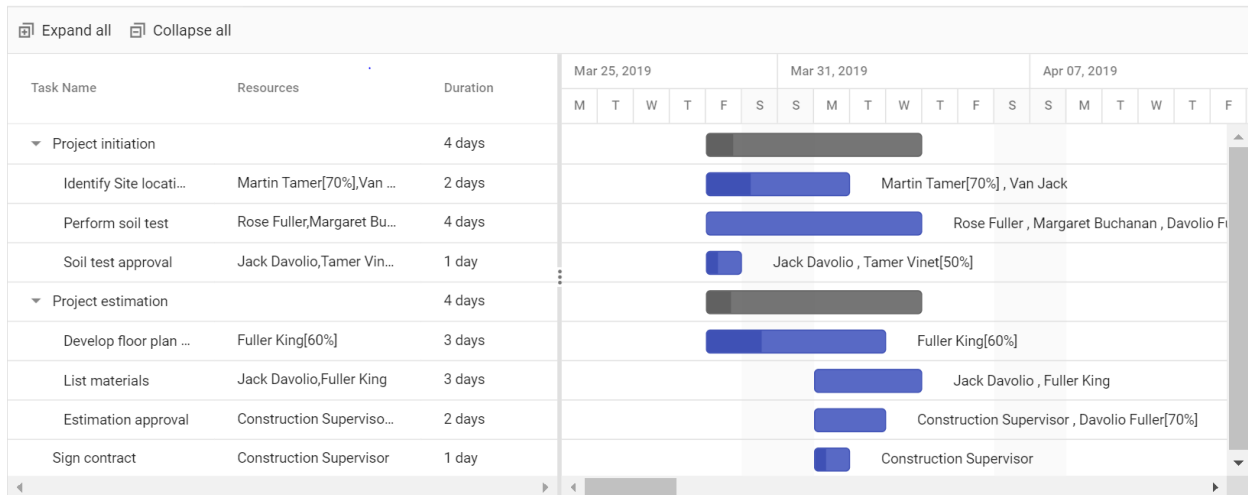
 Progress = 30,
 Work = 60,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 5, Unit = 75 }
 }
 };
 Record2.SubTasks.Add(Record2Child1);
 Record2.SubTasks.Add(Record2Child2);
 Record2.SubTasks.Add(Record2Child3);
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Sign contract",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 1,
 Progress = 30,
 };
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 return GanttDataSourceCollection;
}

public class GanttResources
{
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
 public Nullable<int> Unit { get; set; }
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public List<ResourceModel> Resources { get; set; }
 public int Work { get; set; }
}

public class ResourceModel
{
 public int ResourceId { get; set; }
 public Nullable<int> Unit { get; set; }
}
}

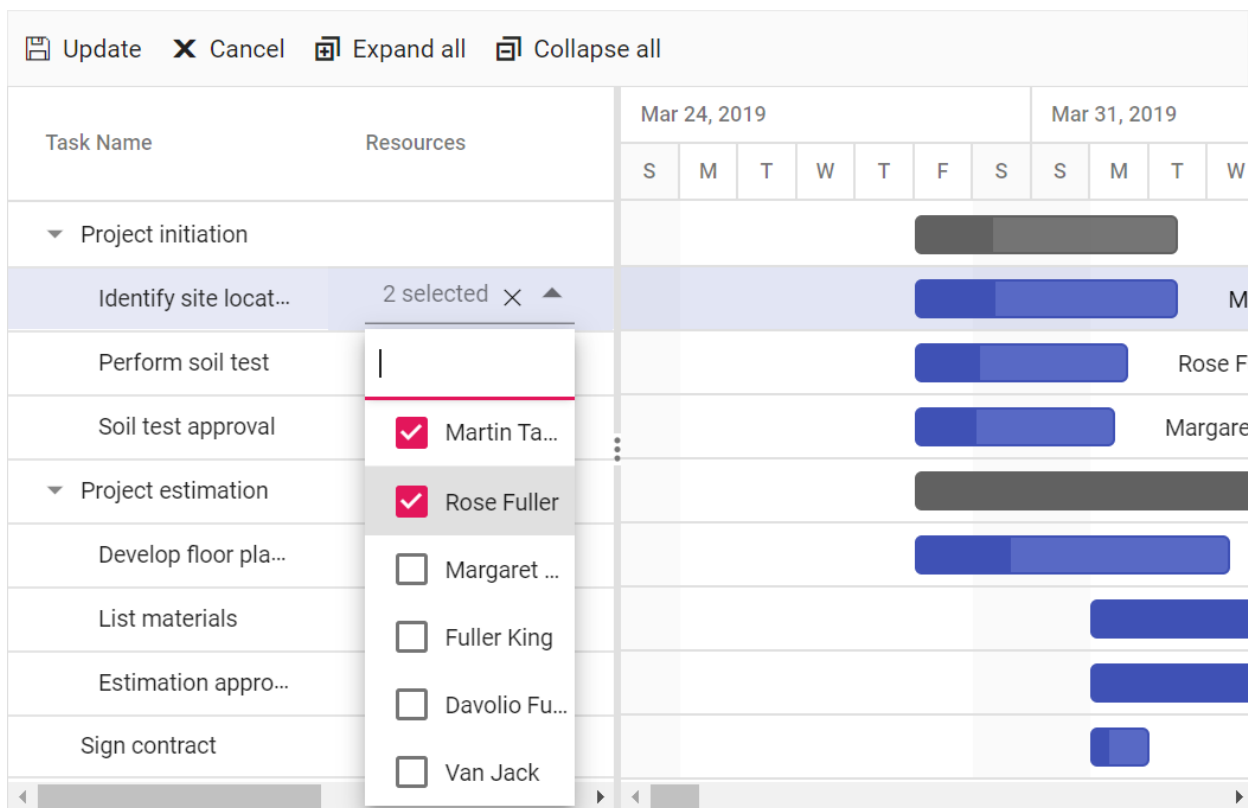
```

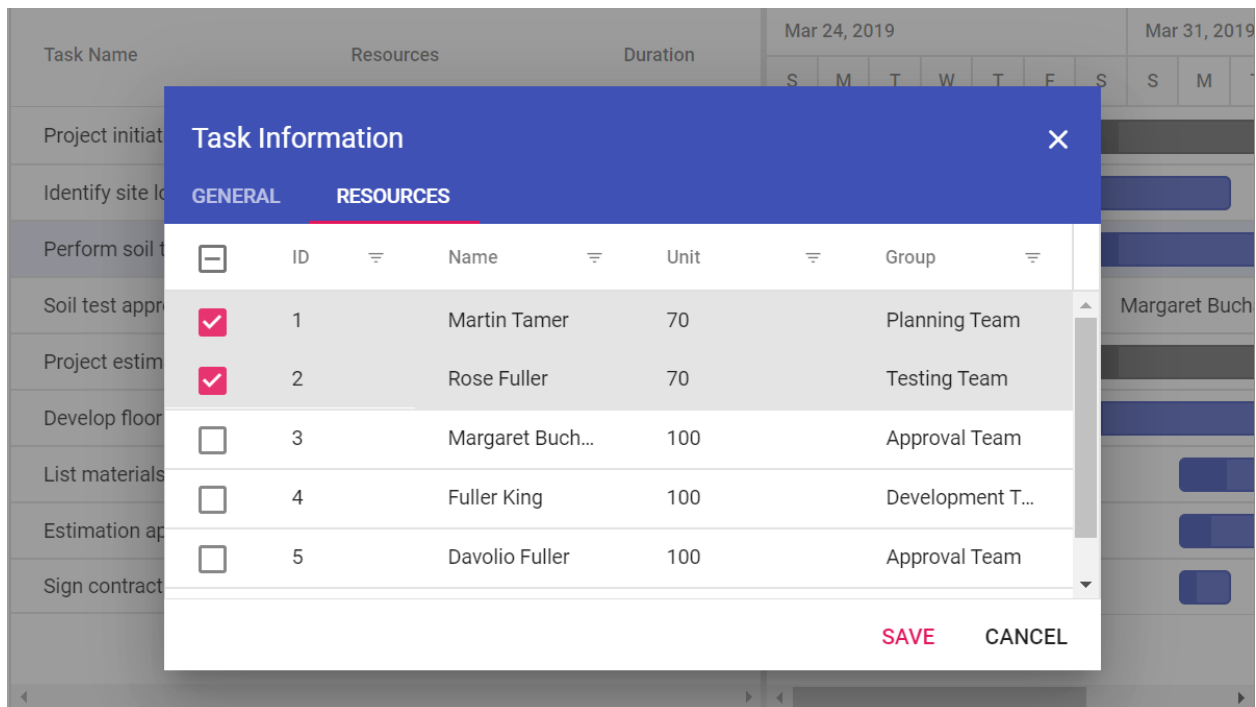


### Add/Edit resource collection

By using cell/ dialog edit option, we can add/remove the multiple resources for a particular task.

Resource Unit can be change for a each task on resource tab in edit dialog by double click on the unit cell.





### Resource View in ASP.NET MVC Gantt Component

The resource breakdown view is used to visualize the tasks assigned to each resource in hierarchy manner. Resources are displayed as parents and all the tasks assigned to each resource are displayed as its child records. It can be initialized by setting the [viewType](#) property to `ResourceView`.

#### Resource task

A task assigned to one or more resources are termed as resource task and it is added as child task to the respective resource. Already assigned task can also be shared or moved with other resources by adding a resource name to the task or removing resource name from the task by cell or dialog editing.

**Note:** Currently there is no support for unscheduled task in Resource view Gantt.

#### CSHTML

```
@(Html.EJS().Gantt("ResourceView").DataSource((IEnumerable<object>
) ViewBag.DataSource).Height("450px").ViewType(Syncfusion.EJ2.Gantt.ViewType.
ResourceView).HighlightWeekends(true).AllowSelection(true).AllowResizing(true)
)
.TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").Work("Work").StartDate("StartDate").EndDate(
"EndDate").Duration("Duration").Progress("Progress")
.ResourceInfo("Resources").Child("SubTasks"))
.TreeColumnIndex(1).ResourceFields(rf =>
rf.Id("ResourceId").Name("ResourceName").Unit("ResourceUnit").Group("Resourc
eGroup"))
.EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true).AllowTaskbarEdit
ing(true).ShowDeleteConfirmDialog(true))
.Toolbar(new List<string>() { "Add", "Edit", "Update", "Delete",
"Cancel", "ExpandAll", "CollapseAll" })
.Resources((IEnumerable<object>) ViewBag.Resources)
```

```

 .Columns(col =>
 {
 col.Field("TaskId").Visible(false).Add();
 col.Field("TaskName").HeaderText("Name").Width(250).Add();
 col.Field("Work").HeaderText("Work").Add();
 col.Field("Progress").Add();
 col.Field("ResourceGroup").HeaderText("Group").Add();
 col.Field("StartDate").Width(100).Add();
 col.Field("Duration").Width(100).Add();
 })
 .LabelSettings(ls =>
ls.RightLabel("Resources")).SplitterSettings(sp => sp.ColumnIndex(3))
 .ProjectStartDate("03/28/2019").ProjectEndDate("05/18/2019")
 .Render()
)

```

### RESOURCEVIEW.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.DataSource = ResourceViewData();
 ViewBag.Resources = GetResourceGroup();
 return View();
 }
 public static List<GanttDataSource> ResourceViewData()
 {
 List<GanttDataSource> GanttResourceSampleCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify Site location",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 2,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 1, ResourceUnit = 50 }
 }
 }
 }
 }
}

```



```

 }
};
GanttDataSource Record1Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 4,
 Progress = 30,
 Work = 20,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 2, ResourceUnit = 70 }
 }
};
GanttDataSource Record1Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 1,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel> {
 new ResourceModel { ResourceId = 3, ResourceUnit = 25 },
 new ResourceModel { ResourceId = 1, ResourceUnit = 75 },
 }
};
Record1.SubTasks.Add(Record1Child1);
Record1.SubTasks.Add(Record1Child2);
Record1.SubTasks.Add(Record1Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record2Child1 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 3,
 Progress = 30,
 Work = 30,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 4, ResourceUnit = 50 },
 new ResourceModel { ResourceId = 2, ResourceUnit = 70 }
 }
};
GanttDataSource Record2Child2 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",

```

```

 StartDate = new DateTime(2019, 04, 01),
 Duration = 3,
 Progress = 30,
 Work = 40,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 6, ResourceUnit = 40 }
 }
 };
 GanttDataSource Record2Child3 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Estimation approval",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 2,
 Progress = 30,
 Work = 60,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 5, ResourceUnit = 75 }
 }
 };
 Record2.SubTasks.Add(Record2Child1);
 Record2.SubTasks.Add(Record2Child2);
 Record2.SubTasks.Add(Record2Child3);
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Sign contract",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 1,
 Progress = 30,
 };
 GanttResourceSampleCollection.Add(Record1);
 GanttResourceSampleCollection.Add(Record2);
 GanttResourceSampleCollection.Add(Record3);
 return GanttResourceSampleCollection;
}

public static List<ResourceGroupCollection> GetResourceGroup()
{
 List<ResourceGroupCollection> GanttResourcesCollection = new
List<ResourceGroupCollection>();
 ResourceGroupCollection Record1 = new ResourceGroupCollection()
 {
 ResourceId = 1,
 ResourceName = "Martin Tamer",
 ResourceGroup = "Planning Team"
 };
 ResourceGroupCollection Record2 = new ResourceGroupCollection()
 {
 ResourceId = 2,
 ResourceName = "Rose Fuller",
 ResourceGroup = "Testing Team"
 };
 ResourceGroupCollection Record3 = new ResourceGroupCollection()
 {
 ResourceId = 3,

```

```

 ResourceName = "Margaret Buchanan",
 ResourceGroup = "Approval Team"
 };
 ResourceGroupCollection Record4 = new ResourceGroupCollection()
 {
 ResourceId = 4,
 ResourceName = "Fuller King",
 ResourceGroup = "Development Team"
 };
 ResourceGroupCollection Record5 = new ResourceGroupCollection()
 {
 ResourceId = 5,
 ResourceName = "Davolio Fuller",
 ResourceGroup = "Approval Team"
 };
 ResourceGroupCollection Record6 = new ResourceGroupCollection()
 {
 ResourceId = 6,
 ResourceName = "Van Jack",
 ResourceGroup = "Development Team"
 };
 GanttResourcesCollection.Add(Record1);
 GanttResourcesCollection.Add(Record2);
 GanttResourcesCollection.Add(Record3);
 GanttResourcesCollection.Add(Record4);
 GanttResourcesCollection.Add(Record5);
 GanttResourcesCollection.Add(Record6);
 return GanttResourcesCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public List<ResourceModel> Resources { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
 public List<IndicatorsModel> Indicators { get; set; }
}

public class ResourceModel
{
 public int ResourceId { get; set; }
 public Nullable<int> ResourceUnit { get; set; }
}

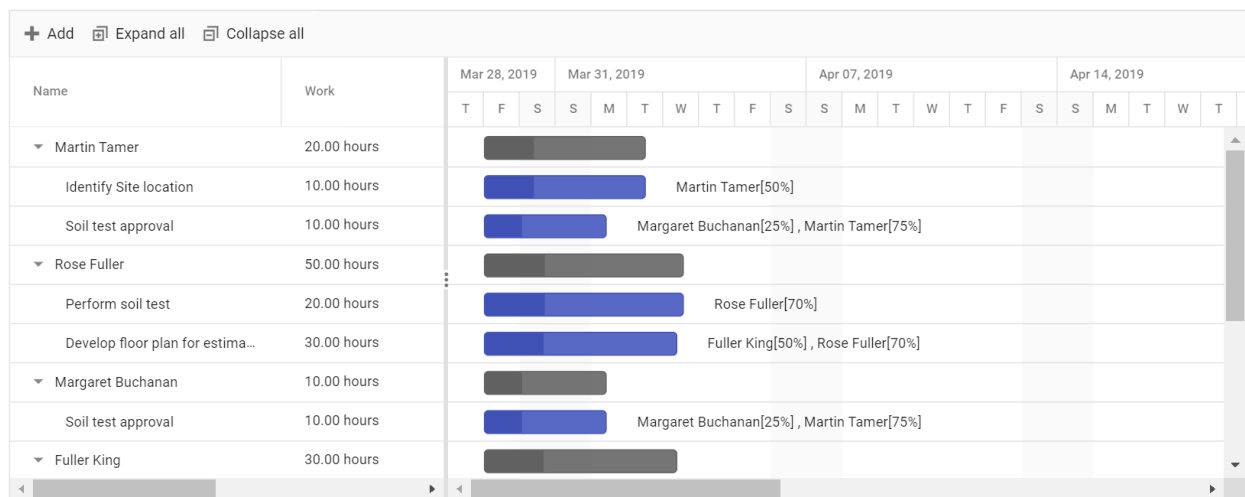
public class IndicatorsModel
{

```

```

 public string date { get; set; }
 public string iconClass { get; set; }
 public string name { get; set; }
 public string tooltip { get; set; }
 }
 public class ResourceGroupCollection
 {
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
 public string ResourceGroup { get; set; }
 }
}

```



### Resource OverAllocation

When a resource is assigned too much of work to complete within a day of resource's available time then it is called as overallocation.

The available working time of resources for completing the task in a day will be calculated based on the `dayWorkingTime` property and `resource unit`.

The range of overallocation dates can be highlighted by a square bracket. It can be enabled by setting the `showOverallocation` property as `true`. The following code example demonstrates how to hide or show the over allocation by clicking the custom button.

**Note:** By default, the `showOverAllocation` property value is `false`.

### CSHTML

```

@ {
 List<object> toolbarItems = new List<object>();
 toolbarItems.Add("Add");
 toolbarItems.Add("Edit");
 toolbarItems.Add("Update");
 toolbarItems.Add("Delete");
 toolbarItems.Add("Cancel");
 toolbarItems.Add("ExpandAll");
}

```

```

 toolbarItems.Add("CollapseAll");
 toolbarItems.Add(new { text = "Show/Hide
Overallallocation", tooltipText = "Show/Hide Overallallocation", id =
"showhidebar" });
 }

@(Html.EJS().Gantt("ShowHideOverAllocation").DataSource((IEnumerable<object>
)ViewBag.DataSource).Height("450px").ViewType(Syncfusion.EJ2.Gantt.ViewType.
ResourceView).HighlightWeekends(true).AllowSelection(true).AllowResizing(tru
e).ShowOverAllocation(true)
 .ToolbarClick("ToolbarClick").Toolbar(toolbarItems)
 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").Work("Work").StartDate("StartDate").EndDate
("EndDate").Duration("Duration").Progress("Progress")

 .Dependency("Predecessor").ResourceInfo("Resources").Child("SubTasks"))
 .TreeColumnIndex(1).ResourceFields(rf =>
rf.Id("ResourceId").Name("ResourceName").Unit("ResourceUnit").Group("Resourc
eGroup"))
 .EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true).AllowTaskbarEdit
ing(true).ShowDeleteConfirmDialog(true))
 .Resources((IEnumerable<object>)ViewBag.Resources)
 .Columns(col =>
 {
 col.Field("TaskId").Visible(false).Add();
 col.Field("TaskName").HeaderText("Name").Width(250).Add();
 col.Field("Work").HeaderText("Work").Add();
 col.Field("Progress").Add();
 col.Field("ResourceGroup").HeaderText("Group").Add();
 col.Field("StartDate").Width(100).Add();
 col.Field("Duration").Width(100).Add();
 })
 .LabelSettings(ls =>
ls.RightLabel("Resources").TaskLabel("Progress"))
 .ProjectStartDate("03/28/2019").ProjectEndDate("05/18/2019")
 .Render()
)
 <script>
 function ToolbarClick(args) {
 if (args.item.id === 'showhidebar') {
 var ganttObj =
document.getElementById("ShowHideOverAllocation").ej2_instances[0];
 ganttObj.showOverAllocation = ganttObj.showOverAllocation ?
false : true;
 }
 };
 </script>

```

### SHOWHIDE.CS

```

public ActionResult Index()
{
 ViewBag.DataSource = ResourceViewData();
 ViewBag.Resources = GetResourceGroup();
}

```

```
 return View();
 }
 public static List<GanttDataSource> ResourceViewData()
 {
 List<GanttDataSource> GanttResourceSampleCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify Site location",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 2,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 1, ResourceUnit = 50 }
 }
 };
 GanttDataSource Record1Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 4,
 Progress = 30,
 Work = 20,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 2, ResourceUnit = 70 }
 }
 };
 GanttDataSource Record1Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 1,
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel> {
 new ResourceModel { ResourceId = 3, ResourceUnit = 25 },
 new ResourceModel { ResourceId = 1, ResourceUnit = 75 },
 }
 };
 Record1.SubTasks.Add(Record1Child1);
 Record1.SubTasks.Add(Record1Child2);
 Record1.SubTasks.Add(Record1Child3);
 GanttDataSource Record2 = new GanttDataSource()
```

```
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Record2Child1 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 3,
 Progress = 30,
 Work = 30,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 4, ResourceUnit = 50 },
 new ResourceModel{ ResourceId = 2, ResourceUnit = 70 }
 }
};
GanttDataSource Record2Child2 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 3,
 Progress = 30,
 Work = 40,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 6, ResourceUnit = 40 }
 }
};
GanttDataSource Record2Child3 = new GanttDataSource()
{
 TaskId = 8,
 TaskName = "Estimation approval",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 2,
 Progress = 30,
 Work = 60,
 Resources = new List<ResourceModel>
 {
 new ResourceModel{ ResourceId = 5, ResourceUnit = 75 }
 }
};
Record2.SubTasks.Add(Record2Child1);
Record2.SubTasks.Add(Record2Child2);
Record2.SubTasks.Add(Record2Child3);
GanttDataSource Record3 = new GanttDataSource()
{
 TaskId = 9,
 TaskName = "Sign contract",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 1,
 Progress = 30,
```

```

 };
 GanttResourceSampleCollection.Add(Record1);
 GanttResourceSampleCollection.Add(Record2);
 GanttResourceSampleCollection.Add(Record3);
 return GanttResourceSampleCollection;
}

public static List<ResourceGroupCollection> GetResourceGroup()
{
 List<ResourceGroupCollection> GanttResourcesCollection = new
List<ResourceGroupCollection>();
 ResourceGroupCollection Record1 = new ResourceGroupCollection()
 {
 ResourceId = 1,
 ResourceName = "Martin Tamer",
 ResourceGroup = "Planning Team"
 };
 ResourceGroupCollection Record2 = new ResourceGroupCollection()
 {
 ResourceId = 2,
 ResourceName = "Rose Fuller",
 ResourceGroup = "Testing Team"
 };
 ResourceGroupCollection Record3 = new ResourceGroupCollection()
 {
 ResourceId = 3,
 ResourceName = "Margaret Buchanan",
 ResourceGroup = "Approval Team"
 };
 GanttResourcesCollection.Add(Record1);
 GanttResourcesCollection.Add(Record2);
 GanttResourcesCollection.Add(Record3);
 return GanttResourcesCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public List<ResourceModel> Resources { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
 public List<IndicatorsModel> Indicators { get; set; }
}

public class ResourceModel
{
 public int ResourceId { get; set; }

```

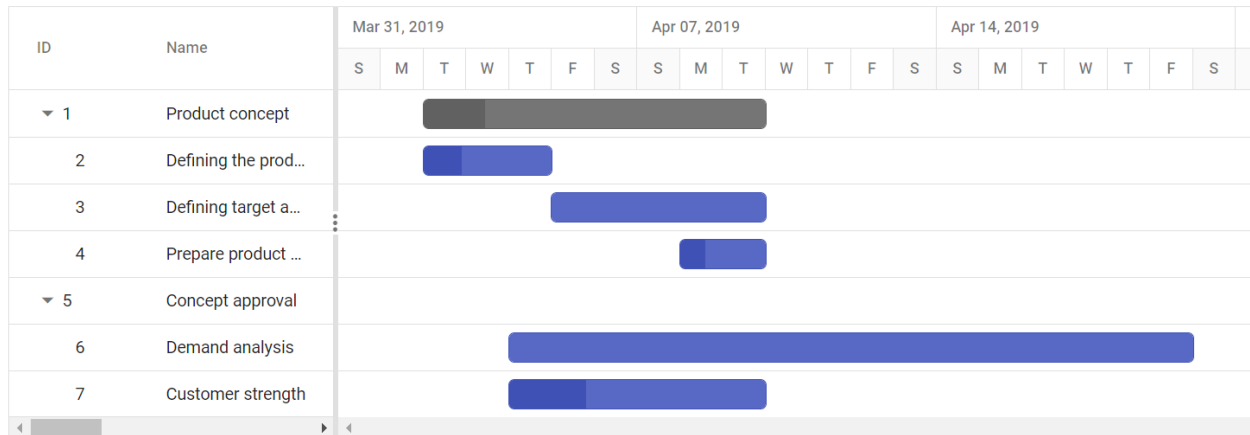


```

 public Nullable<int> ResourceUnit { get; set; }
 }
 public class IndicatorsModel
 {
 public string date { get; set; }
 public string iconClass { get; set; }
 public string name { get; set; }
 public string tooltip { get; set; }
 }
 public class ResourceGroupCollection
 {
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
 public string ResourceGroup { get; set; }
 }

```

Dependency Line Show/Hide ☒



### Unassigned task

A task not assigned to any one of the resource are termed as unassigned tasks. The unassigned tasks are grouped with a name as **Unassigned Task** and displayed at the bottom of Gantt data collection . It is validated at load time during Gantt record creation by default based on a task [resourceInfo](#) mapping property in the Gantt chart data source. If the resource is assigned to the unassigned grouped tasks, the task will be moved as child to the respective resource.

### Enable taskbar drag and drop

In Gantt, you can enable taskbar drag and drop between resources by using the [AllowTaskbarDragAndDrop](#) property. This allows you to move a taskbar from one resource to another vertically, making it easier to schedule tasks and manage resources.

**Note:** By default, the [AllowTaskbarDragAndDrop](#) property value is false.

### CSHTML

```

@ (Html.EJS().Gantt("ResourceMultiTaskbar").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px"))

.ViewType(Syncfusion.EJ2.Gantt.ViewType.ResourceView).HighlightWeekends(true)
.AllowSelection(true).AllowResizing(true).EnableMultiTaskbar(true).AllowTaskbarDragAndDrop(true).ShowOverAllocation(true)

```

```

 .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").Work("Work").StartDate("StartDate").EndDate(
"EndDate").Duration("Duration").Progress("Progress")

 .Dependency("Predecessor").ResourceInfo("Resources").ExpandState("IsExpand")
 .Child("SubTasks"))
 .TreeColumnIndex(1).ResourceFields(rf =>
rf.Id("ResourceId").Name("ResourceName").Unit("ResourceUnit").Group("ResourceGroup"))
 .EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true).AllowTaskbarEditing(true).ShowDeleteConfirmDialog(true))
 .ToolBar(new List<string>() { "ExpandAll", "CollapseAll" })
 .Resources((IEnumerable<object>)ViewBag.Resources)
 .Columns(col =>
{
 col.Field("TaskId").Add();
 col.Field("TaskName").HeaderText("Name").Width(250).Add();
 col.Field("Work").HeaderText("Work").Add();
 col.Field("Progress").Add();
 col.Field("ResourceGroup").HeaderText("Group").Add();
 col.Field("StartDate").Width(100).Add();
 col.Field("Duration").Width(100).Add();
})
 .LabelSettings(ls =>
ls.TaskLabel("TaskName").SplitterSettings(sp => sp.ColumnIndex(2))
 .ProjectStartDate("03/28/2019").ProjectEndDate("05/18/2019")
 .Render()
)
)

```

### TASKBAR DRAG AND DROP.CS

```

public ActionResult Index()
{
 ViewBag.DataSource = MultiTaskbarData();
 ViewBag.Resources = GetResourceGroup();
 return View();
}

public static List<GanttDataSource> MultiTaskbarData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project Initiation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Record1Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify Site Location",
 StartDate = new DateTime(2019, 03, 29),
 Duration = 3,
 };
 Record1.SubTasks.Add(Record1Child1);
 GanttDataSourceCollection.Add(Record1);
 return GanttDataSourceCollection;
}

```

```
 Progress = 30,
 Work = 10,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 1, ResourceUnit = 50 }
 }
 };
 GanttDataSource Record1Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform Soil Test",
 StartDate = new DateTime(2019, 04, 03),
 Duration = 4,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 1, ResourceUnit = 70 }
 },
 Predecessor = "2",
 Progress = 30,
 Work = 20
 };
 GanttDataSource Record1Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil Test Approval",
 StartDate = new DateTime(2019, 04, 09),
 Duration = 4,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 1, ResourceUnit = 25 }
 },
 Predecessor = "3",
 Progress = 30,
 Work = 10
 };
 Record1.SubTasks.Add(Record1Child1);
 Record1.SubTasks.Add(Record1Child2);
 Record1.SubTasks.Add(Record1Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project Estimation",
 StartDate = new DateTime(2019, 03, 29),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
 };
 GanttDataSource Record2Child1 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop Floor Plan for Estimation",
 StartDate = new DateTime(2019, 04, 01),
 Duration = 5,
 Progress = 30,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 2, ResourceUnit = 50 }
 },
 },
```

```

 Work = 50
 };
 GanttDataSource Record2Child2 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List Materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 2, ResourceUnit = 40}
 },
 Predecessor = "6FS-2",
 Progress = 30,
 Work = 40
 };
 GanttDataSource Record2Child3 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Estimation Approval",
 StartDate = new DateTime(2019, 04, 09),
 Duration = 4,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 2, ResourceUnit = 75}
 },
 Predecessor = "7FS-1",
 Progress = 30,
 Work = 60
 };
 Record2.SubTasks.Add(Record2Child1);
 Record2.SubTasks.Add(Record2Child2);
 Record2.SubTasks.Add(Record2Child3);
 GanttDataSource Record3 = new GanttDataSource()
 {
 TaskId = 9,
 TaskName = "Sign Contract",
 StartDate = new DateTime(2019, 04, 09),
 Duration = 1,
 Resources = new List<ResourceModel>
 {
 new ResourceModel { ResourceId = 3 }
 },
 Predecessor = "11FS-5"
 };
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 GanttDataSourceCollection.Add(Record3);
 return GanttDataSourceCollection;
}
public static List<ResourceGroupCollection> MultitaskbarResource()
{
 List<ResourceGroupCollection> GanttResourcesCollection = new
List<ResourceGroupCollection>();
 ResourceGroupCollection Record1 = new ResourceGroupCollection()
 {
 ResourceId = 1,

```

```

 ResourceName = "Martin Tamer",
 ResourceGroup = "Planning Team",
 isExpand = false
 };
 ResourceGroupCollection Record2 = new ResourceGroupCollection()
 {
 ResourceId = 2,
 ResourceName = "Rose Fuller",
 ResourceGroup = "Testing Team",
 isExpand = true
 };
 ResourceGroupCollection Record3 = new ResourceGroupCollection()
 {
 ResourceId = 3,
 ResourceName = "Margaret Buchanan",
 ResourceGroup = "Approval Team",
 isExpand = false
 };
 GanttResourcesCollection.Add(Record1);
 GanttResourcesCollection.Add(Record2);
 GanttResourcesCollection.Add(Record3);
 return GanttResourcesCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string TaskType { get; set; }
 public DateTime? StartDate { get; set; }
 public DateTime? EndDate { get; set; }
 public DateTime BaselineStartDate { get; set; }
 public DateTime BaselineEndDate { get; set; }
 public int? Duration { get; set; }
 public bool IsManual { get; set; }
 public int Progress { get; set; }
 public string Predecessor { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 public List<ResourceModel> Resources { get; set; }
 public string Notes { get; set; }
 public int? Work { get; set; }
 public int ParentID { get; set; }
 public List<IndicatorsModel> Indicators { get; set; }
}

public class ResourceModel
{
 public int ResourceId { get; set; }
 public Nullable<int> ResourceUnit { get; set; }
}

public class IndicatorsModel
{
 public string date { get; set; }
 public string iconClass { get; set; }
 public string name { get; set; }
 public string tooltip { get; set; }
}

public class ResourceGroupCollection

```

```

{
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
 public string ResourceGroup { get; set; }
 public bool isExpand { get; set; }
}

```

### Filtering in gantt control

Filtering allows you to view specific or related records based on filter criteria. This can be done in the Gantt control by using the filter menu support and toolbar search support. To enable filtering in the Gantt control, set the [AllowFiltering](#) to **true**. Menu filtering support can be configured using the [FilterSettings](#) property and toolbar searching can be configured using the [SearchSettings](#) property.

### Filter hierarchy modes

The Gantt supports a set of filtering modes with the [FilterSettings.HierarchyMode](#) property. The following are the types of filter hierarchy modes available in the Gantt control:

- **Parent:** This is the default filter hierarchy mode in Gantt. The filtered records are displayed with its parent records. If the filtered records do not have any parent record, then only the filtered records will be displayed.
- **Child:** Displays the filtered records with its child record. If the filtered records do not have any child record, then only the filtered records will be displayed.
- **Both:** Displays the filtered records with its both parent and child records. If the filtered records do not have any parent and child records, then only the filtered records will be displayed.
- **None:** Displays only the filtered records.

### CSHTML

```

@Html.EJS().DropDownList("FilterMode").DataSource((IEnumerable<object>)ViewBag.dropdata).Width("250px").Value("Parent").Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "id", Value =
"mode" }).Change("onChange").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).AllowFiltering(true).Render()

<script>
 function onChange(e) {
 var mode = e.value;
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 ganttObj.filterSettings.hierarchyMode = mode;
 ganttObj.clearFiltering();
 }
</script>

```

**FILTERHIERARCHYMODE.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 ViewBag.dropdata = new List<object>() {
 new { id= "Parent", mode= "Parent" },
 new { id= "Child", mode= "Child" },
 new { id= "Both", mode= "Both" },
 new { id= "None", mode= "None" },
 };
 return View();
}
```

*Initial filter*

To apply the filter at initial rendering, set the filter to **predicate** object in the [FilterSettings.Columns](#) property.

**CSHTML**

```
@{
 List<object> filterColumns = new List<object>();
 filterColumns.Add(new { field = "TaskName", matchCase = false,
@operator = "startswith", predicate = "and", value = "Identify" });
 filterColumns.Add(new { field = "TaskId", matchCase = false,
@operator = "equal", predicate = "and", value = 2 });
}

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
rogress("Progress").Child("SubTasks")
).AllowFiltering(true).FilterSettings(filter =>
filter.Columns(filterColumns)).Render()
```

**INITIALFILTER.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

*Filter operators*

The filter operator for a column can be defined in the **FilterSettings.Columns.Operator** property.

The available operators and its supported data types are:

Operator	Description	Supported Types
----------	-------------	-----------------

startswith	Checks whether the value begins with the specified value.	String
------------	-----------------------------------------------------------	--------

endswith	Checks whether the value ends with the specified value.	String
----------	---------------------------------------------------------	--------

contains | Checks whether the value contains the specified value. |String

equal | Checks whether the value is equal to the specified value. |String &#124; Number &#124; Boolean &#124; Date

notequal | Checks for the values that are not equal to the specified value. |String &#124; Number &#124; Boolean &#124; Date

greaterthan | Checks whether the value is greater than the specified value. |Number &#124; Date

greaterthanorequal | Checks whether the value is greater than or equal to the specified value. |Number &#124; Date

lessthan | Checks whether the value is less than the specified value. |Number &#124; Date

lessthanorequal | Checks whether the value is less than or equal to the specified value. |Number &#124; Date

---

**Note:** By default, the `FilterSettings.Columns.Operator` value is `equal`

---

### *Diacritics*

By default, the Gantt control ignores the diacritic characters while filtering. To include diacritic characters, set the [FilterSettings.IgnoreAccent](#) to true.

In the following sample, type **Perform** in the **TaskName** column to filter diacritic characters.

### **CSHTML**

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).AllowFiltering(
 true).FilterSettings(filter =>
 filter.IgnoreAccent(true)).Render()
```

### **DIACRITICSFILTER.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
```



```

 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 1 },
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 2, 3, 5 },
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ResourceId = new int[] { 4 },
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 ResourceId = new int[] { 4, 8 },
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);

```

```

 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
 }

 public class GanttDataSource
 {
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
 }

```

### Filtering a specific column by method

You can filter the columns dynamically by using the `filterByColumn` method.

#### CSHTML

```

@Html.EJS().Button("filter").Content("Filter TaskName column").CssClass("e-
primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc
e).Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").Child("SubTasks")
).AllowFiltering(true).Render()

<script>
 document.getElementById('filter').addEventListener('click', function
(args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.filterByColumn('TaskName', 'startswith', 'Iden');
 });
</script>

```

#### FILTERBYCOLUMN.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

### Clear filtered columns

You can clear all the filtering condition done in the Gantt control by using the `clearFiltering` method. The following code snippet explains the above behaviour.

### CSHTML

```
@Html.EJS().Button("clearFilter").Content("Clear Filter").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
).AllowFiltering(true).Render()

<script>
 document.getElementById('clearFilter').addEventListener('click',
function (args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.clearFiltering();
});
</script>
```

### CLEARFILTER.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

## Sorting in ASP.NET MVC Gantt Chart Component

Sorting enables you to sort data in the ascending or descending order. To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the Gantt control, set the [AllowSorting](#) property to true. Sorting options can be configured through the [SortSettings](#) property.

### CSHTML

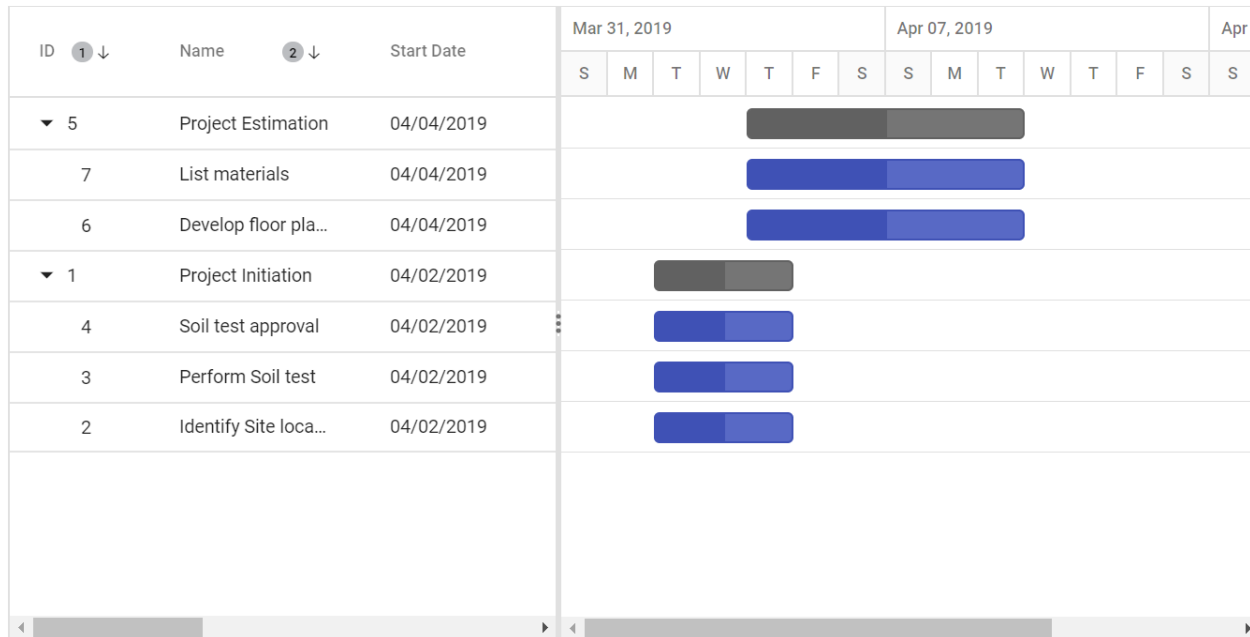
```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).TaskFields(ts => ts.Id("TaskId").Name("TaskName"

).StartDate("StartDate").EndDate("EndDate").Child("SubTasks")).AllowSorting(
true).Render()
```

### ENABLESORTING.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

The following screenshot shows the output of multicolumn sorting in Gantt control.



**Note:** \* Gantt columns are sorted in the ascending order. If you click the already sorted column, the sort direction toggles.

\* To disable sorting for a particular column, set the [Columns.AllowSorting](#) property to false.

#### Sorting column on Gantt initialization

The Gantt control can be rendered with sorted columns initially, and this can be achieved by using the [SortSettings](#) property. You can add columns that are sorted initially in the [SortSettings.Columns](#) collection defined with [Field](#) and [Direction](#) properties. The following code example shows how to add the sorted column to Gantt initialization.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Height("450px").AllowSorting(true).SortSettings(ss=>
 ss.Columns(col=>
 {col.Field("TaskId").Direction(Syncfusion.EJ2.Gantt.SortDirection.Descending)
 }.Add();

 col.Field("TaskName").Direction(Syncfusion.EJ2.Gantt.SortDirection.Ascending)
 }.Add();
)))
 .TaskFields(ts =>
 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
 .Child("SubTasks").Render()
```

#### INITIALSORT.CS

```
public IActionResult Index()
{
```

```
ViewBag.DataSource = GanttData.ProjectNewData();
return View();
}
```

### Sorting column dynamically

Columns in the Gantt control can be sorted dynamically using the `sortColumn` method. The following code example demonstrates how to invoke the `sortColumn` method by clicking the custom button.

#### CSHTML

```
@Html.EJS().Button("Sort").Content("Sort TaskId Column").CssClass("e-
primary").Render()

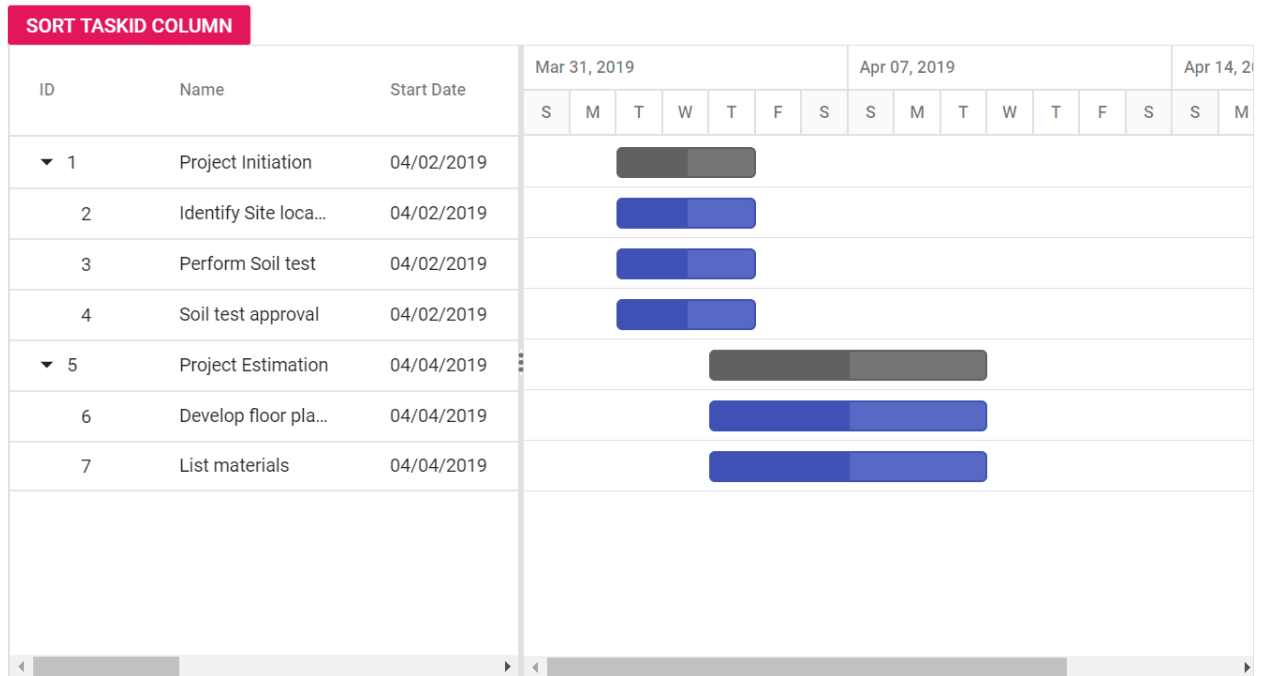
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.TaskFields(ts => ts.Id("TaskId").Name("TaskName"

).StartDate("StartDate").EndDate("EndDate").Child("SubTasks")).AllowSorting(
true).Render()

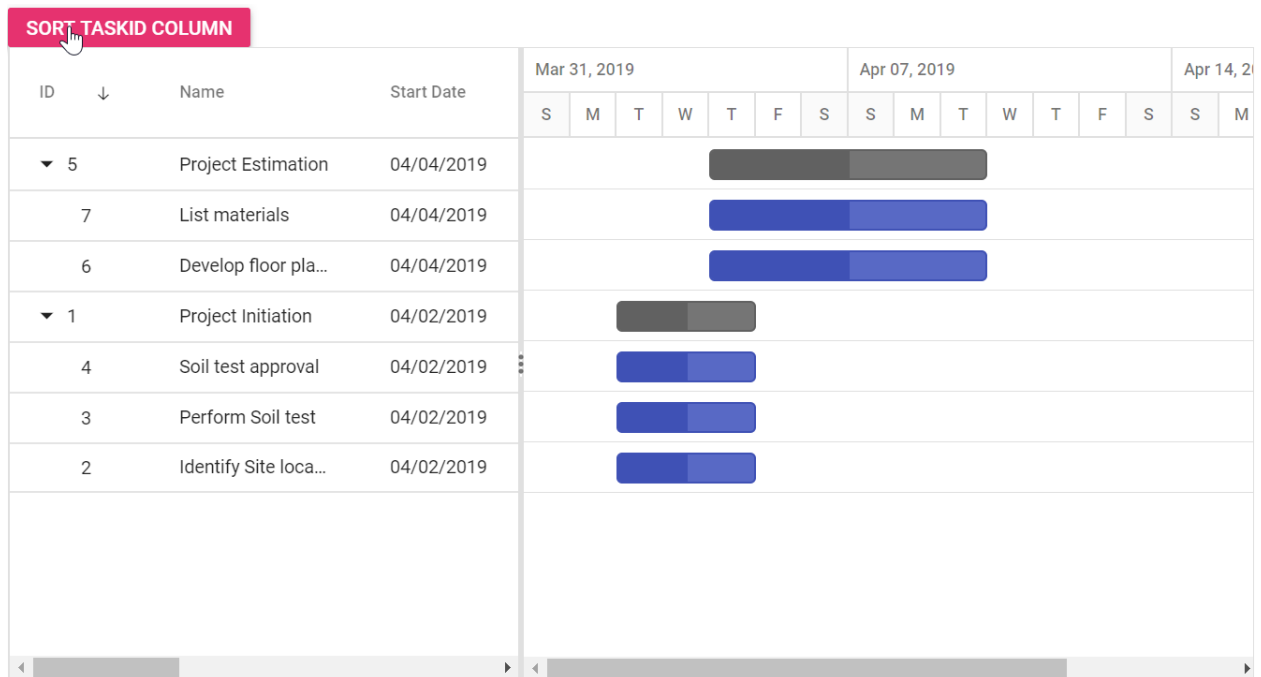
 <script>
 document.getElementById('Sort').addEventListener('click', function
(args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.sortModule.sortColumn('TaskId', "Descending", false)
 });
 </script>
```

#### DYNAMICSORT.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```



Before Sorting



After Sorting

[Clear all the sorted columns dynamically](#)

In the Gantt control, you can clear all the sorted columns and return to previous position using the `clearSorting` public method. The following code snippet shows how to clear all the sorted columns by clicking the custom button.

### CSHTML

```

 @Html.EJS().Button("clearSort").Content("Clear Sorting").CssClass("e-
 primary").Render()

 @Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc
 e).TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate(
 "StartDate").EndDate("EndDate").Child("SubTasks")).AllowSorting(true).Render
 ()

 <script>
 document.getElementById('clearSort').addEventListener('click',
 function (args) {
 var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
 ganttObj.clearSorting();
 });
 </script>

```

### **CLEARSORTING.CS**

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

### Sorting events

During the sort action, the Gantt control triggers two events. The [ActionBegin](#) event triggers before the sort action starts, and the [ActionComplete](#) event triggers after the sort action is completed.

### **CSHTML**

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc
e).TaskFields(ts => ts.Id("TaskId").Name("TaskName"
).StartDate("StartDate").EndDate("EndDate").Child("SubTasks")).ActionBegin("
actionHandler").ActionComplete("actionHandler"
).AllowSorting(true).Render()

<script>
 function actionHandler(args) {
 alert(args.requestType + ' ' + args.type);
 }
</script>

```

### **EVENTHANDLERS.CS**

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

**Note:** The `args.requestType` is the current action name. For example, for sorting the `args.requestType`, value is **sorting**.

### Sorting custom columns

In Gantt, you can sort custom columns of different types like string, numeric, etc., By adding the custom column in the column collection, you can perform initial sort using the `sortSettings` or you can also sort the column dynamically by a button click.

The following code snippets explains how to achieve this.

#### CSHTML

```
@Html.EJS().Button("Sort").Content("Sort Custom Column").CssClass("e-
primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSourc
e).TaskFields(ts => ts.Id("TaskId").Name("TaskName"
).StartDate("StartDate").EndDate("EndDate").Progress("Progress").Child("SubT
asks")).AllowSorting(true).Columns(col =>
{
 col.Field("TaskId").Width(150).Add();
 col.Field("TaskName").HeaderText("Job
Name").Width(250).Add();
 col.Field("StartDate").HeaderText("Star
tDate").Width(250).Add();
 col.Field("EndDate").HeaderText("End
Date").Width(250).Add();

 col.Field("Progress").HeaderText("Progress").Width(250).Add();
 col.Field("CustomColumn").HeaderText("Custom
Column").Width(250).Add();
}).Render()

<script>
 document.getElementById('Sort').addEventListener('click', function
(args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.sortModule.sortColumn('CustomColumn', "Ascending",
false)
 });
</script>
```

#### CUSTOMCOLUMNS.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}

public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
```



```
GanttDataSource Record1 = new GanttDataSource()
{
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 CustomColumn= 2
};
GanttDataSource Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 CustomColumn= 3
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 CustomColumn= 1
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70,
 CustomColumn= 5
};
GanttDataSource Child5 = new GanttDataSource()
{
```

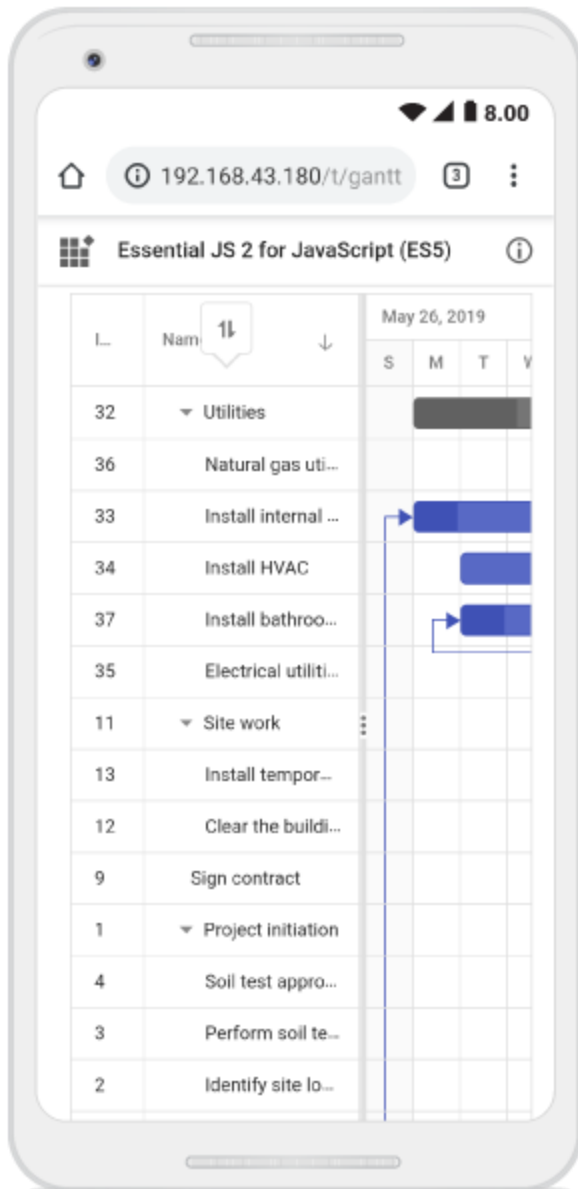
```
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50,
 CustomColumn= 4
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public int CustomColumn { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}
```

### Touch interaction

To perform **tap** action on a column header, trigger [sorting](#) operation to the selected column. A popup is displayed for multi-column sorting. To sort multiple columns, tap the popup, and then tap the desired column headers.

The following screenshot shows Gantt touch sorting,



## Selection

Selection provides an option to highlight a row or a cell. It can be done using arrow keys or by scrolling down the mouse. To disable selection in the Gantt control, set the [AllowSelection](#) to false.

The Gantt control supports two types of selection that can be set by using the [SelectionSettings.Type](#) property. They are:

- **Single:** Sets a single value by default and allows only selection of a single row or a cell.
- **Multiple:** Allows you to select multiple rows or cells. To perform the multi-selection, press and hold the CTRL key and click the desired rows or cells.

### Selection mode

The Gantt control supports three types of selection modes that can be set by using the [SelectionSettings.Mode](#). They are:

- **Row**: Allows you to select only rows, and the row value is set by default.
- **Cell**: Allows you to select only cells.
- **Both**: Allows you to select rows and cells at the same time.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts =>

 ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")

).SelectionSettings(ss=>ss.Mode(Syncfusion.EJ2.Grids.SelectionMode.Both)).Render()
```

### BOTHTYPE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Toggle selection

The toggle selection allows you to select and deselect a specific row or cell. To enable toggle selection, set the **enableToggle** property of the selectionSettings to **true**. If you click the selected row or cell, then it will be deselected and vice versa. By default, the **enableToggle** property is set to **false**.

### CSHTML

```
@Html.EJS().Button("toggle").Content("Disable Toggle").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").AllowSelection(true).SelectionSettings(ss =>

 ss.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple).EnableToggle(true)).TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate(

"StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Render()

 <script>
 document.getElementById('toggle').addEventListener('click', function
 (args) {
 var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
 ganttObj.selectionSettings.enableToggle = false;
 });
 </script>
```

```
});
</script>
```

### TOGGLESELECTION.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Clear selection

You can clear the selected cells and selected rows by using a method called `clearSelection`. The following code example demonstrates how to clear the selected rows in Gantt Chart.

### CSHTML

```
@Html.EJS().Button("selectRows").Content("Select Rows").CssClass("e-
primary").Render()
@Html.EJS().Button("clearSelection").Content("Clear
Selection").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc
e).Height("450px").AllowSelection(true).SelectionSettings(ss =>
 ss.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).TaskFields(ts
=> ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate(
"EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Rend
er()

<script>
 document.getElementById('selectRows').addEventListener('click',
function (args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.selectionModule.selectRows([1, 2, 3]); // passing the
record index as array collection
 });
 document.getElementById('clearSelection').addEventListener('click',
function (args) {
 var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
 ganttObj.clearSelection(); // Clear the selected rows
 });
</script>
```

### CLEARSELECTION.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Get selected row indexes and records

You can get the selected row indexes by using the [getSelectedRowIndexes](#) method. And by using [getSelectedRecords](#) method, you can get the selected record details.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id(
 "TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration(
 "Duration").Progress("Progress").Child("SubTasks")
).SelectionSettings(ss=>ss.Mode(Syncfusion.EJ2.Grids.SelectionMode.Row).Type(
 Syncfusion.EJ2.Grids.SelectionType.Multiple)).RowSelected("rowSelected").Render()

<script>
 function rowSelected(args) {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 var selectedrowindex =
ganttObj.selectionModule.getSelectedRowIndexes(); // get the selected row
indexes.
 alert(selectedrowindex); // to alert the selected row indexes.
 var selectedrecords =
ganttObj.selectionModule.getSelectedRecords(); // get the selected records.
 console.log(selectedrecords); // to print the selected records
in console window.
 }
</script>
```

#### GETSELECTEDROWINDEX.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Multiple selection based on condition

You can select multiple rows based on condition by using the [selectRows](#) method.

In the following code, the rows which contains **TaskId** value as 3 and 4 are selected at initial rendering.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id(
 "TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration(
 "Duration").Progress("Progress").Child("SubTasks")
).SelectionSettings(ss=>ss.Mode(Syncfusion.EJ2.Grids.SelectionMode.Row).Type(
 Syncfusion.EJ2.Grids.SelectionType.Multiple)).DataBound("dataBound").Render()

<script>
 function dataBound(args) {
```

```
var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
var rowIndexes = [];
ganttObj.treeGrid.grid.dataSource.forEach((data, index) => {
 if (data.TaskId === 3 || data.TaskId === 4) {
 rowIndexes.push(index);
 }
});
ganttObj.selectionModule.selectRows(rowIndexes);
}
</script>
```

### **CONDITIONALSELECTION.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

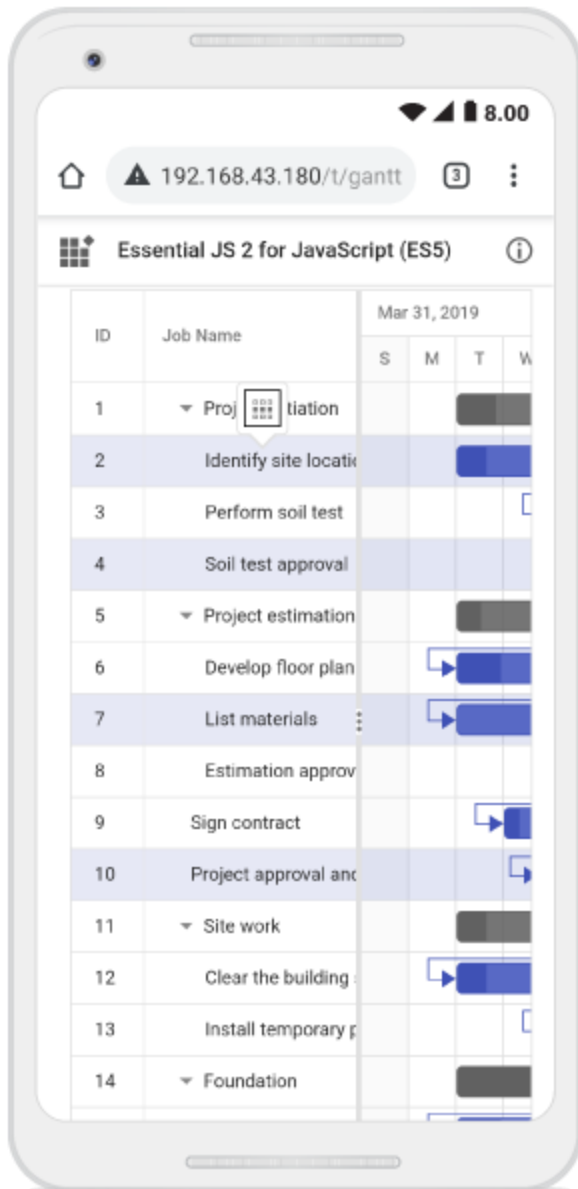
![Alt text](images/conditionalSelection.png)

#### Touch interaction

When you **tap** gantt row, tapped row will be selected.

[Single selection](#) : To select a single row or cell, perform **single tap** on it.

[Multiple selection](#) : To perform multiple selection, **tap** on the multiple selection popup, and then tap the desired rows or cells.



See Also

- [Touch interaction](#)

### Rows in ASP.NET MVC Gantt Component

Row represents a task information from the data source, and it is possible to perform the following actions in Gantt rows.

#### Row height

It is possible to change the height of the row in Gantt by setting row height in pixels to the [RowHeight](#) property. The following code example explains how to change the row height in Gantt at load time.

#### **CSHTML**

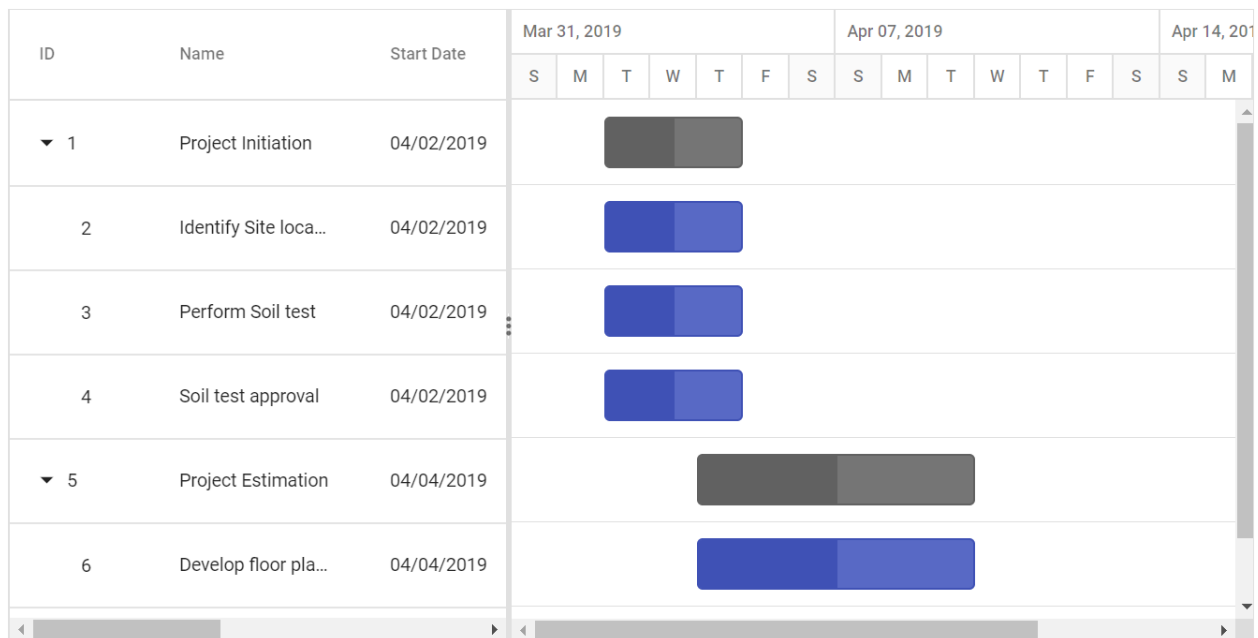


```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").RowHeight(60).TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Render()
```

### ROWHEIGHT.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```



### Expand/Collapse row

In Gantt parent tasks are expanded/collapsed by using expand/collapse icons, expand all/collapse all toolbar items and by using public methods. By default all tasks in Gantt was rendered in expanded state but we can change this status in Gantt.

### Collapse all tasks at gantt chart load

All tasks available in Gantt was rendered in collapsed state by setting [CollapseAllParentTasks](#) property as **true**. The following code example shows how to use this property.

### CSHTML

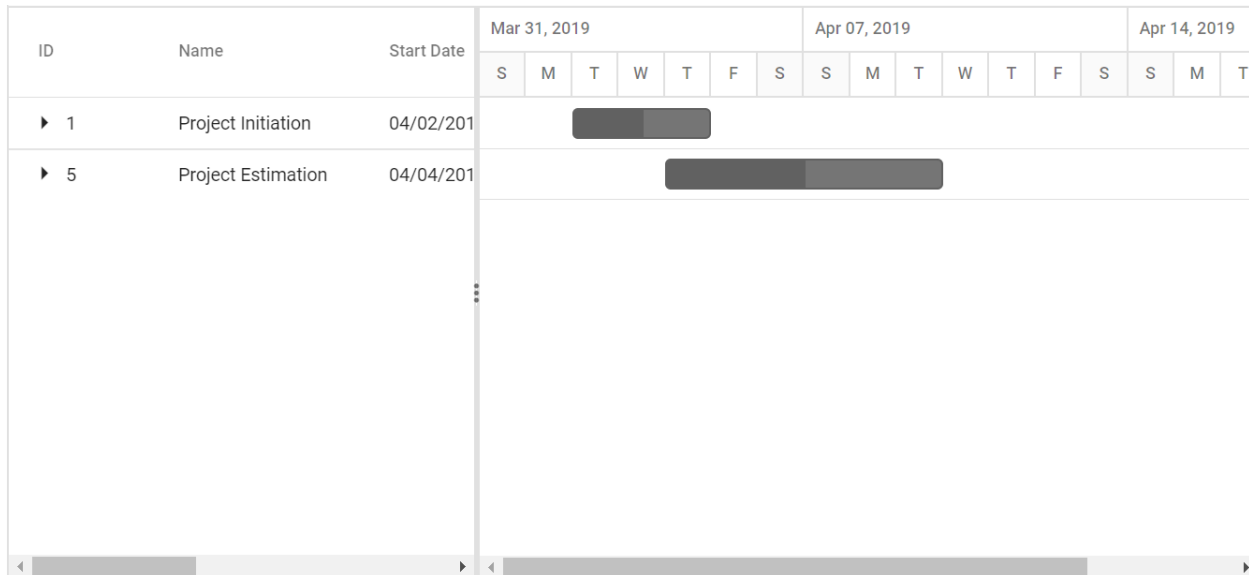
```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").CollapseAllParentTasks(true).TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")
```

```
).Render();
```

### ENABLECOLLAPSEALL.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```



#### Define expand/collapse status of tasks

In Gantt, we can render some tasks in collapsed state and some tasks in expanded state, this can be done by defining expand status of the task in data source. This value was mapped to Gantt control by using [ExpandState](#) property. The following code example shows how to use this property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
.StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
.Child("SubTasks").ExpandState("isExpand")).Render()
```

### EXPANDSTATE.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ProjectNewData();
 return View();
}

public static List<GanttDataSource> ProjectNewData()
```

```
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project Initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 isExpand = true
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify Site location",
 StartDate = new DateTime(2019, 04, 02),
 Progress = 50,
 Duration = 3
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform Soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 3,
 Progress = 50,
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Progress = 50,
 Duration = 3,
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project Estimation",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 0,
 Progress = 50,
 SubTasks = new List<GanttDataSource>(),
 isExpand = false
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
```

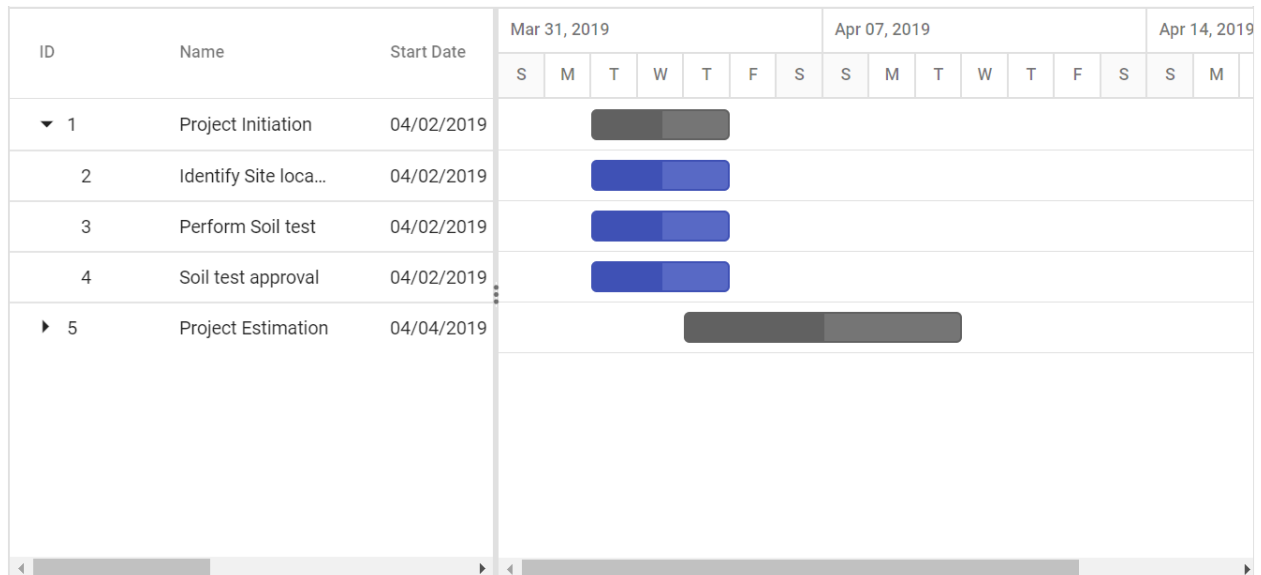
```

 Progress = 50,
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 4,
 Progress = 50,

 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public bool isExpand { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```



#### Customize expand/collapse action

On expand action [Expanding](#) and [Expanded](#) event will be triggered with current expanding row's information. Similarly on collapse action [Collapsing](#) and [Collapsed](#) event will be triggered. Using this events and it's arguments we can customize the expand/collapse action. The following code example

shows how to prevent the particular row from expand/collapse action using [Expanding](#) and [Collapsing](#) event.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").Collapsing("collapsing").Expanding(
 "expanding").TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress(
 "Progress").Child("SubTasks")).Render()

<script>
 function collapsing(args) {
 if (args.data.TaskId == 1) // we can't collapse Task Id 1
 args.cancel = true;
 }
 function expanding(args) {
 if (args.data.TaskId == 5) // we can't expand Task Id 5
 args.cancel = true;
 }
</script>
```

#### EXPANDCOLLAPSEEVENT.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

#### Customize rows

You can customize the appearance of a row in grid side, by using the [rowDataBound](#) event and in chart side by using [queryTaskbarInfo](#) event.

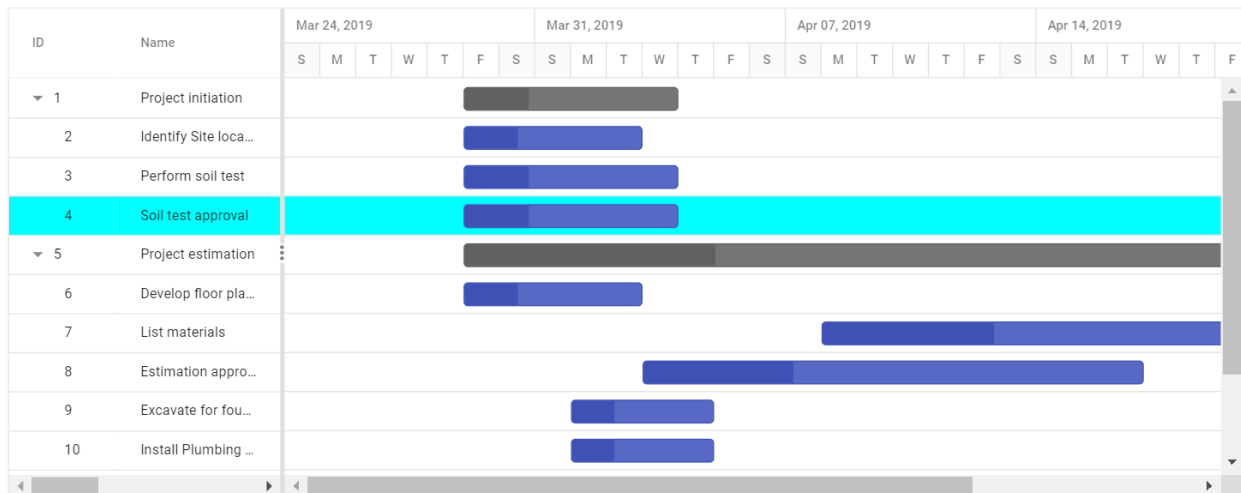
#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).QueryTaskbarInfo(
 "queryTaskbarInfo").RowDataBound("rowDataBound").Render()

<script>
 function rowDataBound(args) {
 if (args.data['TaskId'] == 4) {
 args.row.style.background = 'cyan';
 }
 }
 function queryTaskbarInfo(args) {
 if (args.data['TaskId'] == 4) {
 args.rowElement.style.background = 'cyan';
 }
 }
</script>
```

**CUSTOMIZEROWS.CS**

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

**Styling alternate rows**

You can change the background colour of alternative rows in Gantt chart, by overriding the class as shown below.

```
`css
.e-altrow, tr.e-chart-row:nth-child(even) {
background-color: #f2f2f2;
}
```

**CSHTML**

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

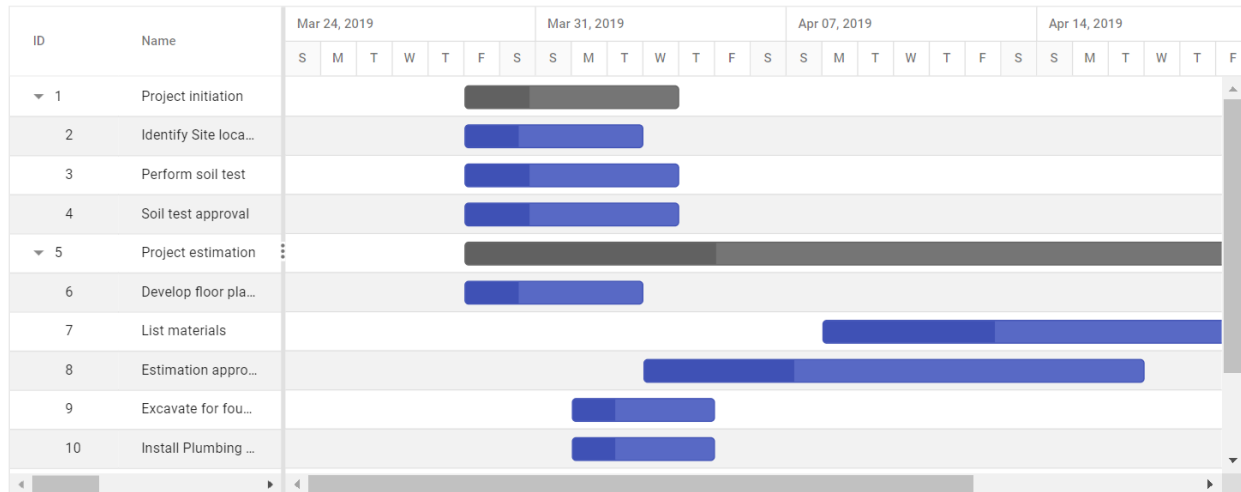
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Render()

<style>
 .e-altrow, tr.e-chart-row:nth-child(even) {
 background-color: #f2f2f2;
 }
</style>
```

**STYLEALTERNATEROWS.CS**

```
public IActionResult Index()
```

```
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```



### Row spanning

Gantt chart has an option to span row cells. You can achieve this using [rowSpan](#) attribute to span cells in the [QueryCellInfo](#) event.

In the following demo, **Soil test approval** cell is spanned to two rows in the **TaskName** column.

### CSHTML

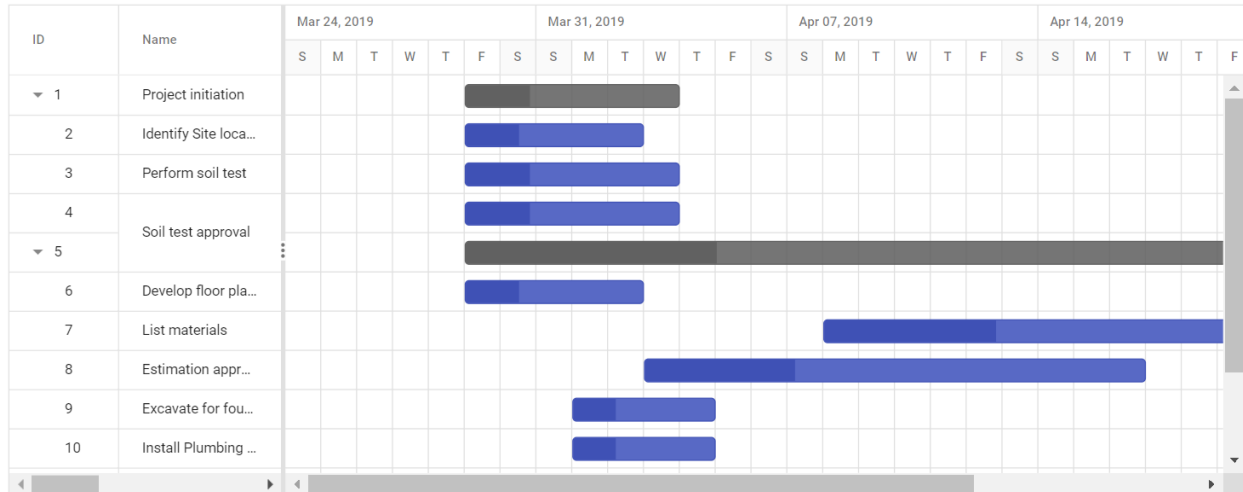
```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks").GridLines(
Syncfusion.EJ2.Gantt.GridLine.Both).QueryCellInfo("queryCellInfo").Render()

<script>
 function queryCellInfo(args) {
 if (args.data['TaskId'] == 4 && args.column.field ===
'TaskName') {
 args.rowSpan = 2;
 }
 }
</script>
```

### ROWSPANNING.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```



### Customize rows and cells

While rendering the TreeGrid part in Gantt, the [RowDataBound](#) and [QueryCellInfo](#) events trigger for every row and cell. Using these events, you can customize the rows and cells. The following code example shows how to customize the cell and row elements using these events.

#### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").QueryCellInfo("queryCellInfo").RowDataBound("rowDataBound").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks")).Render()

<script>
function queryCellInfo(args) {
 if (args.column.field == "Progress") {
 if (args.data.Progress < 60)
 args.cell.style.backgroundColor = "lightgreen"
 else
 args.cell.style.backgroundColor = "yellow"
 }
}
function rowDataBound(args) {
 if (args.data.TaskId == 4)
 args.row.style.backgroundColor = "red"
}
</script>
```

#### CUSTOMIZEROW.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 return View();
}
public static List<GanttDataSource> ganttData()
{
 }
```



```
List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
GanttDataSource Record1 = new GanttDataSource()
{
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Child1 = new GanttDataSource()
{
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
};
GanttDataSource Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
```

```

 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 50
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

ID	Name	Progress	Start Date	Mar 31, 2019							Apr 07, 2019			
				S	M	T	W	T	F	S	S	M	T	V
▼ 1	Project initiation	56	04/02/2019											
2	Identify site locat...	70	04/02/2019											
3	Perform soil test	50	04/02/2019											
4	Soil test approval	50	04/02/2019											
▼ 5	Project estimation	60	04/04/2019											
6	Develop floor pla...	70	04/04/2019											
7	List materials	50	04/04/2019											

### Clip mode

The clip mode provides options to display its overflow cell content and it can be defined by the [Columns.ClipMode](#) property.

The following are three types of **ClipMode**:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when content of the cell overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when content of the cell overflows its area; it displays the tooltip content when hover over ellipsis.

**Note:** By default, all the column's [ClipMode](#) property is defined as `EllipsisWithTooltip`.

### Cell tooltip

You can enable or disable the Grid cell tooltip using the [Columns.ClipMode](#) property.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
.StartDate("StartDate").Duration("Duration").Progress("Progress")
.Child("SubTasks")).Columns(col =>
{
 col.Field("TaskId").Add();

 col.Field("TaskName").Width(100).ClipMode(Syncfusion.EJ2.Grids.ClipMode.EllipsisWithTooltip).Add();
 col.Field("StartDate").Add();

 col.Field("Duration").ClipMode(Syncfusion.EJ2.Grids.ClipMode.Clip).Add();
 col.Field("Progress").Add();
}).Render()
```

### GRIDCELLTOOLTIP.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

## Export

### Excel Export in gantt control

Gantt supports client-side exporting, which allows you to export its data to the Excel and CSV formats.

Use the `excelExport` and `csvExport` methods for exporting. To enable Excel export in the Gantt, set the [AllowExcelExport](#) to true.

### CSHTML

```
@(Html.EJS().Gantt("GanttContainer").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowExcelExport(true).TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate")
.Duration("Duration").Progress("Progress").Child("SubTasks")).Toolbar(new List<string>() {
 "ExcelExport", "CsvExport"
}).ToolbarClick("toolbarClick").Render())
<script>
 function toolbarClick(args) {
 var ganttObj =
document.getElementById("GanttContainer").ej2_instances[0];
 if (args.item.id === "GanttContainer_excelexport") {
 ganttObj.excelExport();
 }
 else if (args.item.id === "GanttContainer_csvexport") {

```

```
 ganttObj.csvExport();
 }
}
</script>
```

## EXCELEXPORT.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = FirstData();
 return View();
}
public static List<GanttDataSource> FirstData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 90
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 40
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 10
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
 GanttDataSource Record2 = new GanttDataSource()
 {
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
```

```

 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child4 = new GanttDataSource()
 {
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 85
 };
 GanttDataSource Child5 = new GanttDataSource()
 {
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 15
 };
 GanttDataSource Child6 = new GanttDataSource()
 {
 TaskId = 8,
 TaskName = "Estimation approval",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 Record2.SubTasks.Add(Child6);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
}

```

## Export

### PDF Export

PDF export allows exporting Gantt data to PDF document. You need to use the `pdfExport` method for exporting. To enable PDF export in the Gantt, set the [allowPdfExport](#) to true.

To export data to PDF document, inject the `PdfExport` module in Gantt.

---

**Note:** Currently, we don't have support for exporting the manually scheduled tasks.

---

### CSHTML

```
@using Syncfusion.EJ2
@{
 ViewBag.Title = "DefaultFunctionalities";
}
<div class="container body-content">

@(Html.EJS().Gantt("GanttContainer").DataSource((IEnumerable<object>)ViewBag
.DataSource).AllowPdfExport(true).TaskFields(ts => ts.Id(
"TaskId").Name("TaskName").StartDate("StartDate").Duration("Duration").Progr
ess("Progress").Child("SubTasks")).Toolbar(new List<string>() {
 "PdfExport" }).ToolbarClick("toolbarClick").Render())
<script>
 function toolbarClick(args) {
 var ganttObj =
document.getElementById("GanttContainer").ej2_instances[0];
 if (args.item.id === "GanttContainer_pdfexport") {
 ganttObj.pdfExport();
 }
 }
</script>
</div>
```

### PDFEXPORT.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}
```

### Indicators in PDF exporting

The PDF export functionality allows users to export Gantt charts enriched with dynamic indicators and accompanying images.

These indicators, represented by images, can be effortlessly defined using the `base64` encoding value in the data object of datasource. This data object field should be mapped to indicator property of [task fields](#).

### CSHTML

```
@using Syncfusion.EJ2
@{
 ViewBag.Title = "DefaultFunctionalities";
}
<div class="container body-content">

@(Html.EJS().Gantt("GanttContainer").DataSource((IEnumerable<object>)ViewBag
.DataSource).AllowPdfExport(true).TaskFields(ts => ts.Id(
"TaskId").Name("TaskName").StartDate("StartDate").Duration("Duration").Progr
ess("Progress").Child("SubTasks")).Toolbar(new List<string>() {
 "PdfExport"
})).ToolbarClick("toolbarClick").Indicators("indicators").Render())
<script>
```

```
function toolbarClick(args) {
 var ganttObj =
document.getElementById("GanttContainer").ej2_instances[0];
 if (args.item.id === "GanttContainer_pdfexport") {
 ganttObj.pdfExport();
 }
}
</script>
</div>
```

**PDFEXPORT.CS**

[illegible]

rNCBFEiR1p4ADbOlKUC1KUC1KUC1KUC1KUC1Kj09UPxQbrw8dh8fHsJvdwsGqGsMtyz2W4QHTY1W  
iAx2zuM1p3tGC0IDjEYErTogs5Hmi6mF5BFJx5uJHmXHk3443t20JsFwyPFMNYCZacfj2u4pJHOL  
onNp26krbqxEhtNtOrHfJeTcdyS+46APE2zPTwo+FLpzwNuTOFYW0t0yC6q3KyrKpMcW5+SzBFU  
QyRFLtR2+oxZjoSi0JEqqbrjrrmgfDofLM23aGXHcjMLKs5Nq5ahgYzAcjyo8m02NJCUG86hqLb  
nnOMxnm1RskRhhhhWVSSYDM/QKUpQKUpQfJJ8fz1/JdLrHs1tkTJb7UaJFAnXnnDQG2gFOZERL8E  
RERVVV+rlX9vOsI8SDD77qBw/ta7JjLUqTkV2we8RLaxGJUeekHCdEABU+PURKiJ/8qlUZjEmmFa  
8c4hZg4cXxK0mdImYj+trS59V7sywvPXbI3muS3xhlztOXW243KcgCSKqLyIxBwxRU/6gAhX6xUk  
+NblbcNyOD7uNGrLqBpzkLKcQyFtXYNXiiYC6gkoEJAYibZiQqJAYiQqioqItURXWUHiEkXmiqi  
p+dKtc+khwm9YhwhrQ/do0qNHvmS3042xHxIe7GUm2kMEVE+arjTvJU+C/Ffz1sw8OJw5tPONOLN  
j3mmJWsRznTxwnj/En1KJ9VKqWlKUoFKUoPGa/aa4dum0cyHT7UDH4GVYblMRVYvzkWVvqQHNCFU  
IVQ23AMRMHAUTbMAMCEXekq7a56Xaj+lb4x1rynFoGQZRp073XLBiuzxQY2c2N5oEl2992MstlIi  
uOCiqYchfjxJSxkbNpsrXlam8aLhnX+K1sPyDTFqbb7RlcWUzfsUuc9X/ABbfdI/WIq6LRIqtusO  
yI5EouI2klXUacNsBoM37V9y+LbydueGao4VLWXjOb2pm6w+pl1x6L1j9JGf7RuNjIZcQ2XQEy7b  
rTgKvMVRIdV4fSKcRXKdG9x2TbQNS5F/hMzfMk41aLu28D2NXiGTztytiMKYrjHdbR98xdbcbbadg  
uogK7KJVsPUC1KUC1KUC1KUC1KUCqqe7C+SPUKepGiYGl9t7OC1kDuGWibbrk838mrP5UmXIhy2  
o5A65KBqfKYJwHRQ5jTauE0CElgrjRbl/tReFXrrnTcvILfcImKybVa51kd7M633C4KNuhSW3OsF  
b7UqUy4piXWAgRChEiCsWfoldo0ePiusWvU5i3PS5UpnAbK8El9JUMGwan3EXGeSMq26rtrUDVSN  
FjOonbFV7gTtYpidrwLFrZYrHa7fZrHZYjUC32+BHCNFgR2gQGmWmgRabbABEREURBRERERertKU  
oFKUoFKUoFFTmlK433uy0pfXyRVqu94rWZtyhMRrOkMOZdw7dv+oOZPZHf9DNhb5kMh7vvXS4YXb  
ZM11znldZPGypqXNVXmq8+a1123wWbTBajRWWo8aOAttNNggA0CJyQRRPgiIiIiI1VrNQfWjbiYu  
olxasemmi8KwhLJI0adDucuY2z1KiCb4TGgM+X/cjIpz+PT+ap5OGpu4n77Ni2nGrNzs0XH7lmdt  
KTKgRXieZYdB5xk+giRC6SVtSRF5qKFy5ly6lspSZwvXXhGsbCnNOL8MX279TPNkfXSisLKUC1KU  
ClKUfAL1TWyW+8PfiNYTuo0uj+xW/OLrHvfmRbe25GsWWwDB7uk346Rh8oW2pQi8Trkh9q4madKc  
qsFbKNz9s3p7StOdV7OMBmJn2PxLu5Fh3ELi3bJDjaLIhK+IihuR3+6wFzRVDZNCESRRTWH1JW0S  
Pu94QGqbCMW4r3pzETP7RImSn2G4Z20TdlkiNivcc031PZAHBUFN8FXoVEcDUH0U25666g7SNWdJ  
56XCTF0zyCHELZLkXE322I91beQoTLBDyYbbfGpvr0lyM5zi9IkhEYTX0pSgUpSgUpSgUpSgUpSghQ9b  
TrXa7Hsw0c06ej3Ar3lWaPZJefBsFitx7bBdjvg4Skho4R3WOoIgKioDvMhVBQtz/AE5m2n7WDg4  
aLW5+Jj7N1y21FmU+Tamun3D3N0pkVyQagBOSAguw2DUkXp8dAEiAAWooV6a5/KDdNohpotr7K4  
lis3J/clk9Xl+5y0jdtCdHa9o6uvrXr8jl0j0czse7fNfLXtt0FwfTqxP3CXZMAX+BjdvfnGBY  
no8OM3Han0gEAVxQbFSUQFFV5CifCg9pSlKBSlKBSlKAvlpXBN/A3f0V/wDqudfRsuCb+Bu/o/y  
rNmvt4TTqhQhzD8bJ368v3lq4f6d38jFoT+yJP8AHyap4Zh+Nk79eX7ylcP9O5+RiK0/ZEn+Pk1  
6GW7KfNdpRn+/n9bw3WpSlUBSlKBSlKBSlKDK8rx0157ilzsV8tdvvnjvUR2BcLfPjhJiz47oKDr  
LrRooONmBEJCSKhIqoqKi1WX9L5/cZ4/WZ6PZb98Mml2rJdOAlWh09AS42+W3MedU3e2545N2mSg  
F2+tsNpFAUULCz5VYPUR+4N6yCN8jvvl77qrbfI95+m6PlVGy9x6012uXb94k9jnz6ehnr7vIusL  
PlKUoFKUoFKUoFKUoKwPrVvypuA/+Krd/+veKs/VWz9bTopdLFvO0c1FekW8rJlWFvY3EYAzWU3I  
ts52Q+bgqKAjZBdY6AqGqgo08xFEFSsE7T9dPtoNq+mmpaWv2P7ImK2vJ/bfJ8n2/zYjUnsd3oDu  
dHd6evoHq6efSnPlQZDpSlApSlApSlAX60rgm/gbv6P8AKudfrSuCb+Bu/o/yrNmvt4TTqhQhzD  
8bJ368v3lq4f6dz8jFoV+yJP8fJqnhmH42Tv15fvLVw/07n5GLQR9kSf4+TXoZbsp812lGf7+f1v  
DdalKVQFKUoFKUoFKUoFVguJX/wCskxz/AMq6cfwliqz7VYPUR+/z6yCN8jvjb7FqrbfI95+h6/k  
rGY9x6013efc9nk9jny6utnr7XMugLP1KUoFKUoFKUoFKUoIT/W06KWu+7MNHNRXpFwG94rmj2Nx  
GAMEiuR7lBdkPm4Kipq4J2qOgKhoiIbvMSVRUD0PTmb1vtn+DhotcX5ePvXXERUWGz41qd6vb/bH  
ShxW5AKzk3IOClDfNCVOrYEMREDBK9bxyNsN13icJrXHA7EtX09SrB7xb40C3HcZVykW2QzcmoTT  
AEhm5JOILA9PNUV5CQTVogow/RLbuo8jFdYtBZz1uZlxZTOFWvkiR/lTAcBqBcSce5qyjbStWtAB  
UE1WS6qdwUXthPfsLKBSlKBSlKAvlpXBN/A3f0f5VzL9aVwzfwn39H+VZ819VvCadUKEOYfjZO/X  
l+8tXD/TufkYtCv2RJ/j5NU8Mw/Gyd+vL95auH+nc/IxaE/siT/Hya35bsp812lH+h38/reG61KU  
qgKUpQKUpQKUpQdZleWWvA8Wud9vltz9msdkioZ7hcJ8gI0WBHaBTdeddNUBtsAEiIiVEFEVVES  
qw3plo1032+oKvust9mW+y3u3RMmlJuEGDCMos2RcXFhOxm1tTZbE7uTgkSuryYQF5qfWMzfSt  
3kfaFwgNU30ftw3vUaImAWiPMivvtzDuQmlLFFaV024FvGe8BuEgIbAivWqo2eopPotsN10+2kas  
6st1uEaLqZkEozJy7JitxsNvx7U28pTWXyLk+24/PfYXpHkBWxE6iJSEAmvpSlApSlApSlApSlAq  
nuwscj09fQromeJY7e9go5A7mdohW62sMt/Jq8eVGLx4cRqQINORQdnxWBCNoVog04rYtGgrasqM  
71Q/C+uvE02Hx8hwmYXC/6oaPS3LxZbfAa+Nvd4D/AGwuMJprugCuKDBeKeQOumsFGWh6n15hI9i  
mWWvPMWtl9sdzt95sd7iNT7fcIEgJMWfHdBdaeadBVBxswISEhVUJFRUVUWuzqGD0hnFambkNDLj  
tvzB5Xsm0jtQz8YnuPypEm7WNZCtmy6poTbfguPRmW0RwUVh9hsGkSMZlM/QKUpQKUpQFTnXy42j  
oKi/nTktfVFTmlc2iJjSeQiZzb0b+1Tmc3nXlridZ7IxNklIG1wL9BKJHRS6u2CvwnHuhOfL5zpF  
y/wC7n8akh2sbZ8U2d7fsX0ywmNLiYxiETw4ISpBSHLHqIyM3C+JERmZLy5IilyRBRERMiIvMfjX  
4Kr8K6izinojki/zv7luMvqlKUSUpSgUpSgUpWrnF/wCJJYUffsgyPUy5okjIJC1ZMQgFCclM3K+  
PMPORGN0A2+mOPZcddJXAXtMuICq6TYEEIvqmt7V94hHEawnavpdI99t+D3WPZPDi3BtuNfctnmD



[illegible]

```

 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>>();
 List<IndicatorsData> Indicators = new List<IndicatorsData>>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>>();
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 Indicators = new List<IndicatorsModel>>() {

```

```

 new IndicatorsModel() {date = "04/08/2019",
iconClass="e-btn-icon e-notes-info e-icons e-icon-left e-gantt e-notes-
info::before", name= "Custom String", tooltip="Follow up",base64 =
indicatorImage},

 new IndicatorsModel() {date = "04/11/2019",
iconClass="e-btn-icon e-notes-info e-icons e-icon-left e-gantt e-notes-
info::before", name= "String Template" base64
= indicatorImage}
 }
};
GanttDataSource Child2 = new GanttDataSource()
{
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
};
GanttDataSource Child3 = new GanttDataSource()
{
 TaskId = 4,
 TaskName = "Soil test approval",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50
};
Record1.SubTasks.Add(Child1);
Record1.SubTasks.Add(Child2);
Record1.SubTasks.Add(Child3);
GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70,
 Indicators = new List<IndicatorsModel>() {
 new IndicatorsModel() {date = "04/10/2019",
iconClass="e-btn-icon e-notes-info e-icons e-icon-left e-gantt e-notes-
info::before", name= "Indicator title", tooltip="tooltip",base64 =
indicatorImage}
 }
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,

```

```

 Progress = 50
 };
 Record2.SubTasks.Add(Child4);
 Record2.SubTasks.Add(Child5);
 GanttDataSourceCollection.Add(Record1);
 GanttDataSourceCollection.Add(Record2);
 return GanttDataSourceCollection;
}

public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public List<IndicatorsModel> Indicators { get; set; }
}

public class IndicatorsModel
{
 public string date { get; set; }
 public string iconClass { get; set; }
 public string name { get; set; }
 public string tooltip { get; set; }
 public string base64 { get; set; }
}

```

### Exporting Gantt data as a blob object

In Gantt, you can export the Gantt chart data as a blob object, which allows you to preview or modify the data before exporting it.

To export the Gantt chart data as a blob object, follow these steps:

step 1: pdfExport fourth argument set as `true`.

step 2: Then , `pdfExpComplete` return as blob object.

### CSHTML

```

@(Html.EJS().Gantt("GanttContainer").DataSource((IEnumerable<object>)ViewBag
.DataSource).AllowPdfExport(true).AllowExcelExport(true).TaskFields(ts =>
ts.Id(

"TaskId").Name("TaskName").StartDate("StartDate").Duration("Duration").Progr
ess("Progress").Child("SubTasks").Toolbar(new List<string>() {
"PdfExport", "ExcelExport"
}).ToolbarClick("toolbarClick").PdfExportComplete("pdfExpComplete").ExcelExp
ortComplete("excelExpComplete").TreeColumnIndex(1).Render())
<script>
 function toolbarClick(args) {
 var ganttObj =
document.getElementById("GanttContainer").ej2_instances[0];
 if (args.item.id === "GanttContainer_pdfexport") {
 ganttObj.pdfExport(null, null, null, true);
 }
 }

```

```

 if (args.item.id === "GanttContainer_excelexport") {
 ganttObj.excelExport(null, null, null, true);
 }
 }
 let excelExpComplete = (args) => {
 //This event will be triggered when excel exporting.
 args.promise.then((e) => {
 //In this `then` function, we can get blob data through the
 arguments after promise resolved.
 exportBlob(e.blobData);
 });
 };
 let pdfExpComplete = (args) => {
 //This event will be triggered when pdf exporting.
 args.promise.then((e) => {
 //In this `then` function, we can get blob data through the
 arguments after promise resolved.
 exportBlob(e.blobData);
 });
 };
 let exportBlob = (blob) => {
 let a = document.createElement('a');
 document.body.appendChild(a);
 a.style.display = 'none';
 let urlg = window.URL.createObjectURL(blob);
 a.href = url;
 a.download = 'Export';
 a.click();
 window.URL.revokeObjectURL(url);
 document.body.removeChild(a);
 }
</script>

```

### **BLOB-DATA.CS**

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.ProjectNewData();
 return View();
}

```

#### *Single page exporting in gantt*

In Gantt, we have provided support to export the Gantt component where each rows are auto-fit to the PDF document page width by setting `IsFitToWidth` as true in `FitToWidthSettings` of `PdfExportProperties`.

Also, we can customize the chart width and grid width in exported file using `ChartWidth` and `GridWidth` by defining it as percentage in string.

### **CSHTML**

```

@using Syncfusion.EJ2
@{
 ViewBag.Title = "DefaultFunctionalities";
}

```

```

<div class="container body-content">

@ (Html.EJS () .Gantt ("GanttContainer") .DataSource ((IEnumerable<object>) ViewBag
.DataSource) .AllowPdfExport (true) .TaskFields (ts => ts.Id (

"TaskId") .Name ("TaskName") .StartDate ("StartDate") .Duration ("Duration") .Progr
ess ("Progress") .Child ("SubTasks") .Toolbar (new List<string> () {
 "PdfExport" }) .ToolBarClick ("toolbarClick") .Render ())

<script>
 function toolbarClick (args) {
 var ganttObj =
document.getElementById ("GanttContainer") .ej2_instances [0];
 if (args.item.id === "GanttContainer_pdfexport") {
 var exportProperties = {
 fitToWidthSettings: {
 isFitToWidth: true,
 }
 };
 ganttObj.pdfExport (exportProperties);
 }
 }
</script>
</div>

```

### **SINGLE-PAGE-EXPORT.CS**

```

public IActionResult Index ()
{
 ViewBag.DataSource = GanttData.ProjectNewData ();
 return View ();
}

```

#### *Exporting with column template*

The PDF export functionality allows to export Grid columns that include images, hyperlinks, and custom text to an PDF document using [pdfQueryCellInfo](#) event.

In the following sample, the hyperlinks and images are exported to PDF using [hyperlink](#) and [image](#) properties in the [pdfQueryCellInfo](#) event.

Note: PDF Export supports base64 string to export the images.

### **CHTML**

```

@Html.EJS () .Gantt ("Gantt") .DataSource ((IEnumerable<object>) ViewBag.DataSource
e) .AllowPdfExport (true) .Height ("450px") .TaskFields (ts
=>

ts.Id ("TaskId") .Name ("TaskName") .StartDate ("StartDate") .EndDate ("EndDate") .D
uration ("Duration") .Progress ("Progress") .Child (
 "SubTasks") .Toolbar (new List<string> () {
 "PdfExport"
 }) .ToolBarClick ("toolbarClick") .PdfQueryCellInfo ("pdfQueryCellInfo") .Resourc
eInfo ("ResourceId") .ResourceFields (rf =>
 rf.Id ("ResourceId") .Name ("ResourceName") .Resources ((
 IEnumerable<object>) ViewBag.projectResources) .Columns (col =>
 {

```

```

 col.Field("TaskId").HeaderText("Task ID").Width(250).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(250).Add();

col.Field("ResourceId").HeaderText("Resources").Template("#columnTemplate").
Add();

 col.Field("StartDate").Add();
 col.Field("Duration").Add();
 col.Field("Progress").Add();
 }).Render()
</script>
 function toolbarClick(args) {
 var ganttObj =
document.getElementById("GanttContainer").ej2_instances[0];
 if (args.item.id === "GanttContainer_pdfexport") {
 ganttObj.pdfExport();
 }
 }
 function pdfQueryCellInfo(args) {
 if (args.column.headerText === 'Resources') {
 {
 args.image = { height: 40, width: 40,
base64: (args).data.taskData.resourcesImage };
 }
 }
 }
</script>
<script type="text/x-jsrender" id="columnTemplate">
 ${if(ganttProperties.resourceNames)}
 <div class="image">
 <div
style="display:inline-block;width:100%;position:relative;left:30px;top:-
14px">${ganttProperties.resourceNames}</div>
 </div>
 ${/if}
</script>

```

### PDF-COLUMNTEMPLATE.CS

```

public IActionResult Index()
{
 ViewBag.DataSource = ganttData();
 ViewBag.projectResources = projectResources();
 return View();
}

public static List<GanttResources> projectResources()
{
 List<GanttResources> GanttResourcesCollection = new
List<GanttResources>();
 GanttResources Record1 = new GanttResources()
 {
 ResourceId = 1,
 ResourceName = "Martin Tamer"
 };
 GanttResources Record2 = new GanttResources()
 {
 ResourceId = 2,

```

```
 ResourceName = "Rose Fuller"
 };
 GanttResources Record3 = new GanttResources()
 {
 ResourceId = 3,
 ResourceName = "Margaret Buchanan"
 };
 GanttResources Record4 = new GanttResources()
 {
 ResourceId = 4,
 ResourceName = "Fuller King"
 };
 GanttResources Record5 = new GanttResources()
 {
 ResourceId = 5,
 ResourceName = "Davolio Fuller"
 };
 GanttResources Record6 = new GanttResources()
 {
 ResourceId = 6,
 ResourceName = "Van Jack"
 };
 GanttResources Record7 = new GanttResources()
 {
 ResourceId = 7,
 ResourceName = "Fuller Buchanan"
 };
 GanttResources Record8 = new GanttResources()
 {
 ResourceId = 8,
 ResourceName = "Jack Davolio"
 };
 GanttResources Record9 = new GanttResources()
 {
 ResourceId = 9,
 ResourceName = "Tamer Vinet"
 };
 GanttResources Record10 = new GanttResources()
 {
 ResourceId = 10,
 ResourceName = "Vinet Fuller"
 };
 GanttResources Record11 = new GanttResources()
 {
 ResourceId = 11,
 ResourceName = "Bergs Anton"
 };
 GanttResources Record12 = new GanttResources()
 {
 ResourceId = 12,
 ResourceName = "Construction Supervisor"
 };
 GanttResourcesCollection.Add(Record1);
 GanttResourcesCollection.Add(Record2);
 GanttResourcesCollection.Add(Record3);
 GanttResourcesCollection.Add(Record4);
 GanttResourcesCollection.Add(Record5);
```

```

GanttResourcesCollection.Add(Record6);
GanttResourcesCollection.Add(Record7);
GanttResourcesCollection.Add(Record8);
GanttResourcesCollection.Add(Record9);
GanttResourcesCollection.Add(Record10);
GanttResourcesCollection.Add(Record11);
GanttResourcesCollection.Add(Record12);
return GanttResourcesCollection;
}
public static List<GanttDataSource> ganttData()
{
 List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
 GanttDataSource Record1 = new GanttDataSource()
 {
 TaskId = 1,
 TaskName = "Project initiation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
 };
 GanttDataSource Child1 = new GanttDataSource()
 {
 TaskId = 2,
 TaskName = "Identify site location",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 70,
 ResourceId = new int[] { 1 },
 };
 GanttDataSource Child2 = new GanttDataSource()
 {
 TaskId = 3,
 TaskName = "Perform soil test",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 Notes = "Obtain an engineered soil test of lot where
construction is planned.From an engineer or company specializing in soil
testing",
 ResourceId = new int[] { 2 },
 };
 GanttDataSource Child3 = new GanttDataSource()
 {
 TaskId = 4,
 TaskName = "Soil test approval",
 Dependency = "3FS",
 Notes = "Measure the total property area allotted for
construction",
 StartDate = new DateTime(2019, 04, 02),
 Duration = 4,
 Progress = 50,
 ResourceId = new int[] { 3 }
 };
 Record1.SubTasks.Add(Child1);
 Record1.SubTasks.Add(Child2);
 Record1.SubTasks.Add(Child3);
}

```



```

GanttDataSource Record2 = new GanttDataSource()
{
 TaskId = 5,
 TaskName = "Project estimation",
 StartDate = new DateTime(2019, 04, 02),
 EndDate = new DateTime(2019, 04, 21),
 SubTasks = new List<GanttDataSource>(),
};
GanttDataSource Child4 = new GanttDataSource()
{
 TaskId = 6,
 TaskName = "Develop floor plan for estimation",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Progress = 70,
 ResourceId = new int[] { 4 },
};
GanttDataSource Child5 = new GanttDataSource()
{
 TaskId = 7,
 TaskName = "List materials",
 StartDate = new DateTime(2019, 04, 04),
 Duration = 3,
 Dependency = "6FS",
 Progress = 50,
 ResourceId = new int[] { 3 },
};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
public class GanttResources
{
 public int ResourceId { get; set; }
 public string ResourceName { get; set; }
}
public class GanttDataSource
{
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public string Dependency { get; set; }
 public string Notes { get; set; }
 public DateTime StartDate { get; set; }
 public DateTime EndDate { get; set; }
 public int? Duration { get; set; }
 public int Progress { get; set; }
 public List<GanttDataSource> SubTasks { get; set; }
 public int[] ResourceId { get; set; }
}

```

### Context menu in ASP.NET MVC Gantt component

The Gantt control allows you to perform quick actions by using context menu. When right-clicking the context menu, the context menu options are shown. To enable this feature, set the

`enableContextMenu` to true. The default context menu options are enabled using the `editSettings` property. The context menu options can be customized using the `contextMenuItems` property.

The default items are listed in the following table.

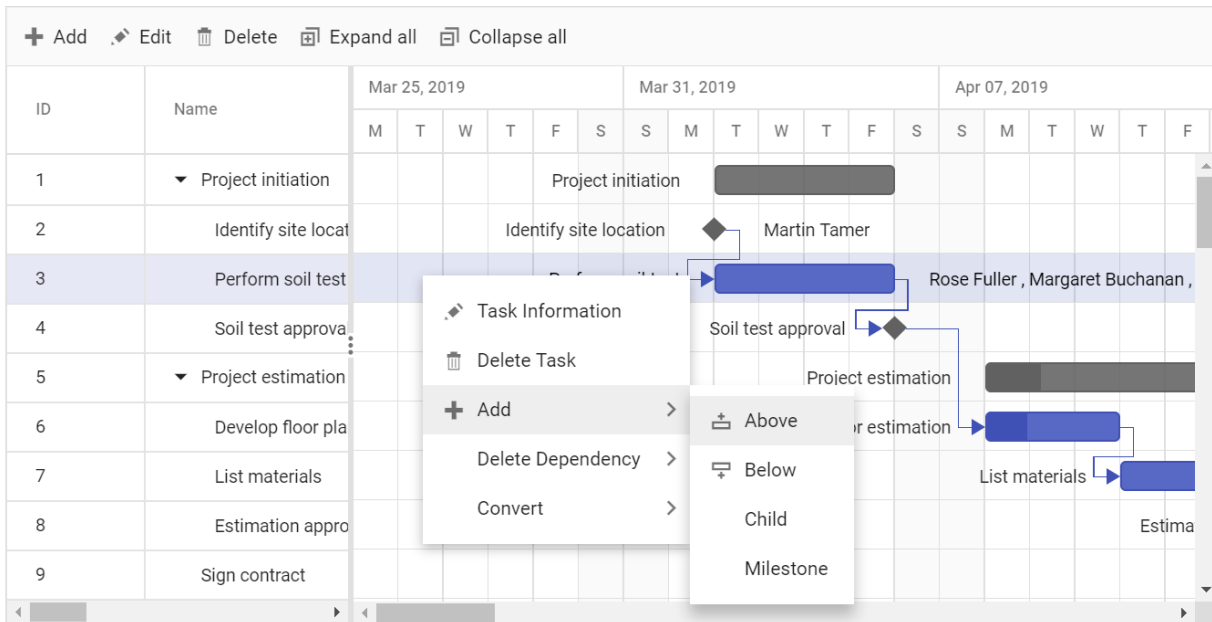
Items	Description
----	----
<code>AutoFit</code>	Auto-fits the current column.
<code>AutoFitAll</code>	Auto-fits all columns.
<code>SortAscending</code>	Sorts the current column in ascending order.
<code>SortDescending</code>	Sorts the current column in descending order.
<code>TaskInformation</code>	Edits the current task.
<code>Add</code>	Adds a new row to the Gantt.
<code>Indent</code>	Indent the selected record to one level.
<code>Outdent</code>	Outdent the selected record to one level.
<code>DeleteTask</code>	Deletes the current task.
<code>Save</code>	Saves the edited task.
<code>Cancel</code>	Cancel the edited task.
<code>DeleteDependency</code>	Deletes the current dependency task link.
<code>Convert</code>	Converts current task to milestone or vice-versa.

### CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
 "TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
 rogress("Progress").Dependency("Predecessor").Child("SubTasks")
).EnableContextMenu(true).AllowSorting(true).AllowResizing(true).EditSetting
 s(es => es.AllowAdding(true).AllowEditing(
 true).AllowDeleting(true)).Render()
```

### CONTEXTMENU.CS

```
public IActionResult Index()
{
 ViewBag.DataSource = GanttData.EditingData();
 return View();
}
```



### Custom context menu items

The custom context menu items can be added by defining the [contextMenuItems](#) as a collection of [\[contextMenuItemModel\]](#). Actions for the customized items can be defined in the [contextMenuClick](#) event and the visibility of customized items can be defined in the [\[contextMenuOpen\]](#) event.

To create custom context menu items for header area, define the target property as [.e-gridheader](#).

The following sample shows context menu item for parent rows to expand or collapse child rows in the content area and a context menu item to hide columns in the header area.

### CSHTML

```
@{
 List<object> contextItems = new List<object> { "AutoFitAll",
 "AutoFit", "TaskInformation", "DeleteTask", "Save", "Cancel",
 "SortAscending", "SortDescending", "Add", "DeleteDependency",
 "Convert" };
 contextItems.Add(new { text = "Collapse the Row", target = ".e-
content", id = "collapserow" });
 contextItems.Add(new { text = "Expand the Row", target = ".e-
content", id = "expandrow" });
 contextItems.Add(new { text = "Hide Column", target = ".e-
gridheader", id = "hidecols" });
}

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc
e).Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName"

).StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("P
rogress").Dependency("Predecessor").Child("SubTasks")

).EnableContextMenu(true).AllowSorting(true).AllowResizing(true).EditSetting
s(es => es.AllowAdding(true).AllowEditing(true).AllowDeleting(true))
```

```

).ContextMenuItems(contextItems).ContextMenuOpen("ContextMenuOpen").ContextM
enuClick("ContextMenuClick").Render()
<script>
 function ContextMenuOpen(args) {
 var record = args.rowData;
 if (args.type !== 'Header') {
 if (!record.hasChildRecords) {
 args.hideItems.push("Collapse the Row");
 args.hideItems.push("Expand the Row");
 } else {
 if (record.expanded) {
 args.hideItems.push("Expand the Row");
 } else {
 args.hideItems.push("Collapse the Row");
 }
 }
 }
 }

 function ContextMenuClick(args) {
 var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
 var record = args.rowData;
 if (args.item.id === 'collapserow') {
 ganttObj.collapseByID(Number(record.ganttProperties.taskId));
 }
 if (args.item.id === 'expandrow') {
 ganttObj.expandByID(Number(record.ganttProperties.taskId));
 }
 if (args.item.id === 'hidecols') {
 ganttObj.hideColumn(args.column.headerText);
 }
 }
}
</script>

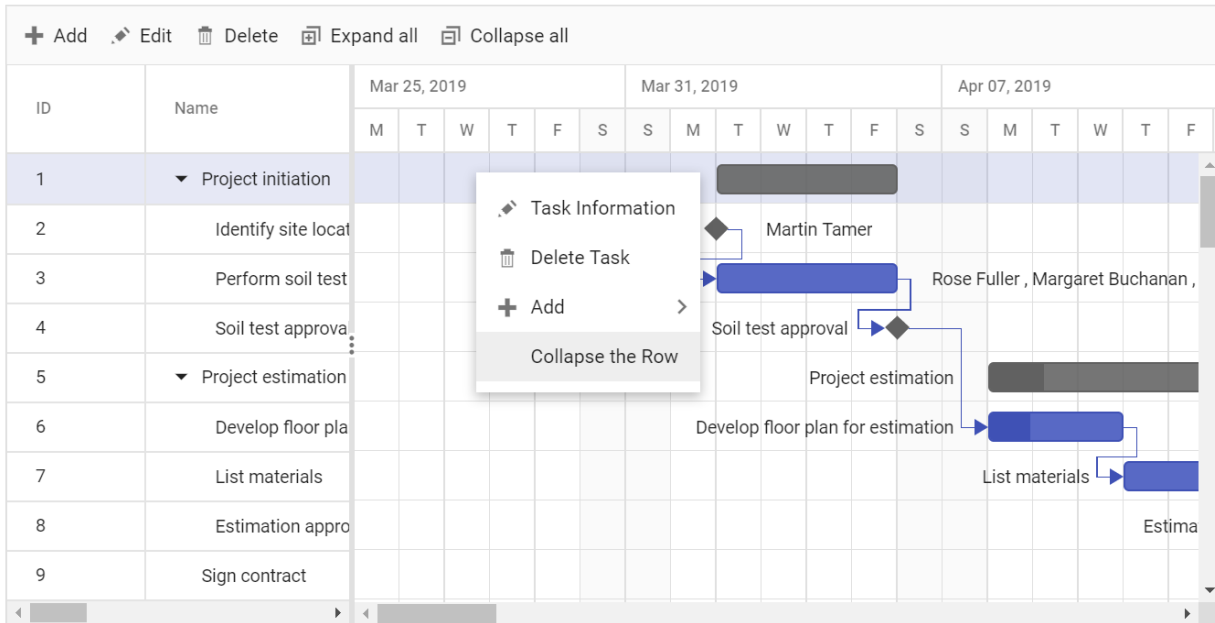
```

### **CUSTOMCONTEXTMENU.CS**

```

public IActionResult Index()
{
 ViewBag.DataSource = GanttData.EditingData();
 return View();
}

```



**Note:** You can show an specific item in context menu for header/content area in the Gantt control by defining the `target` property.

### Touch interaction

To perform **long press** action on a row, [context menu](#) is opened, and then tap a menu item to trigger its action.

### State Persistence Feature

State persistence refers to the Gantt's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores gantt's model object in the local storage when the [enablePersistence](#) is defined as true.

### Get or set localStorage value

If the [enablePersistence](#) property is set to true, the gantt property value is saved in the **window.localStorage** for reference. You can get/set the localStorage value by using the `getItem/setItem` method in the **window.localStorage**.

``typescript`

`//get the Gantt model.`

`var value = window.localStorage.getItem('ganttGantt'); // "ganttGantt" is component name + component id.`

`var model = JSON.parse(model);`

```

``typescript`

`//set the Gantt model.`

window.localStorage.setItem('ganttgantt', JSON.stringify(model)); //"ganttgantt" is component name + component id.

Prevent columns from persisting

When the [EnablePersistence](#) property is set to true, the Gantt properties such as [Filtering](#), [Sorting](#), and [Columns](#) will persist. You can use the `addOnPersist` method to prevent these Gantt properties from persisting.

The following example demonstrates how to prevent Gantt columns from persisting. In the [DataBound](#) event of the Gantt, you can override the `addOnPersist` method and remove the columns from the key list given for persistence.

Note: When the [EnablePersistence](#) property is set to true, the Gantt properties such as column template, column formatter, header text, and value accessor will not persist.

CSHTML

```
@Html.EJS().Button("add").Content("Add Columns").IsPrimary(true).Render()
@Html.EJS().Button("remove").Content("Remove Columns").IsPrimary(true).Render()
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).DataBound("onBound").EnablePersistence(true).Columns(col =>
{
    col.Field("TaskID").HeaderText("Task ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("TaskName").HeaderText("Task Name").Width("170").Add();
    col.Field("StartDate").HeaderText("Start Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Duration").HeaderText("Duration").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).Render()
<script>
    function onBound(args) {
        var cloned = this.addOnPersist;
        this.addOnPersist = function (key) {
            key = key.filter(item => item !== "columns");
            return cloned.call(this, key);
        };
    }
    document.getElementById('add').onclick = function () {
        var obj = { field: "Progress", headerText: 'Progress', width: 100 };
        document.getElementById('Gantt').ej2_instances[0].columns.push(obj);
        //you can add the columns by using the Grid columns method

        document.getElementById('Gantt').ej2_instances[0].treeGrid.refreshColumns();
    }
    document.getElementById('remove').onclick = function () {
        document.getElementById('Gantt').ej2_instances[0].columns.pop();

        document.getElementById('Gantt').ej2_instances[0].treeGrid.refreshColumns();
    }
</script>
```

COLUMN-PREVENT.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = GanttData.EditingData();
    return View();
}
```

Persist the header template and header Text

By default, the Gantt properties such as column template, header text, header template, column formatter, and value accessor will not persist when [EnablePersistence](#) is set to true. Because the column template and header text can be customized at the application level.

If you wish to restore all these column properties, then you can achieve it by cloning the gantt's columns property using JavaScript Object's assign method and storing this along with the persist data manually. While restoring the settings, this column object must be assigned to the gantt's columns property to restore the column settings as demonstrated in the following sample.

CSHTML

```
@Html.EJS().Button("restore").Content("Restore").IsPrimary(true).Render()
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.dataSource).EnablePersistence(true).Columns(col =>
{
    col.Field("TaskID").HeaderText("Task ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("TaskName").HeaderText("Task Name").Width("170").HeaderTemplate("#customertemplate").Add();
    col.Field("StartDate").HeaderText("Start Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Duration").HeaderText("Duration").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).Render()
<script>
document.getElementById('restore').onclick = function () {
    var savedProperties =
JSON.parse(document.getElementById('Gantt').ej2_instances[0].getPersistData());
    var gridColumnState = Object.assign([],
document.getElementById('Gantt').ej2_instances[0].gantColumns);
    savedProperties.columns.forEach(function (col) {
        var headerText = gridColumnState.find(function (colColumnsState) {
return colColumnsState.field === col.field; })['headerText'];
        var colTemplate = gridColumnState.find(function (colColumnsState) {
return colColumnsState.field === col.field; })['template'];
        var headerTemplate = gridColumnState.find(function
(colColumnsState) { return colColumnsState.field === col.field;
})['headerTemplate'];
        col.headerText = 'Text Changed';
        col.template = colTemplate;
        col.headerTemplate = headerTemplate;
    });
};
```

```

        console.log(savedProperties);

document.getElementById('Gantt').ej2_instances[0].TreeGrid.setProperties(savedProperties);
}
</script>

<script id="customertemplate" type="text/x-template">
    <span class="e-icons e-header" ></span>
    Task Name
</script>

<style>
    .e-Reply:before {
        content: '\e815';
    }

    .e-header:before {
        content: '\ea9a';
    }
</style>

```

COLUMN-PERSIST.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = GanttData.EditingData();
    return View();
}

```

Accessibility

The Gantt component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Gantt component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |


```
| Mobile Device Support |  |  
  
| Keyboard Navigation Support |  |  
  
| Accessibility Checker Validation |  |  
  
| Axe-core Accessibility Validation |   
  
|  
  
<style>  
.post .post-content img {  
display: inline-block;  
margin: 0.5em 0;  
}  
</style>  
  
<div> - All  
features of the component meet the requirement.</div>  
  
<div> - Some features of the component do not meet the requirement.</div>  
  
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Gantt component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Gantt component:

The following ARIA attributes are used in Gantt:

- | Attributes | Purpose |
|---------------------|--|
| grid (role) | This attribute is added to the e-table element present in the Gantt, which represents Grid part |
| gridcell (role) | This attribute is added to the td elements present within the e-table, which represents the work cells of Gantt |
| columnheader (role) | This attribute is added to the th elements within the e-table, which represents the header cells of Grid table |
| separator (role) | This attribute is added to the e-split-bar element, which represents the splitter between the Grid table and Chart |
| dialog (role) | This attribute is added to the e-dialog element, which represents the pop-up dialog |

| **toolbar (role)** | This attribute is added to the **e-gantt-toolbar** element, which represents the toolbars of Gantt |

| **aria-label** | It indicates the element's information
 It is assigned to the Gantt UI elements such as timeline cell, taskbar, left label, right label, dependency line, and event markers. |

| **aria-selected** | This attribute is assigned to the Gantt chart row, and it defaults to **false**. The value is changed to **true** when the user selects a grid cell or task |

| **aria-expanded** | This attribute is assigned to the Gantt chart parent task row. The value is changed to **true** when the user clicks a parent taskbar to expand. After the user clicked a parent taskbar to collapse, the attribute value is changed to **false** |

| **aria-grabbed** | This attribute is assigned to the taskbars of Gantt when the user tries to achieve taskbar editing |

Keyboard navigation

The Gantt component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Gantt component.

| **Press** | **To do this** |

| --- | --- |

| **Home** | Selects the first row. |

| **End** | Selects the last row. |

| **DownArrow** | Moves the cell focus/row or cell selection downward. |

| **UpArrow** | Moves the cell focus/row or cell selection upward. |

| **LeftArrow** | Moves the cell focus/row or cell selection left side. |

| **RightArrow** | Moves the cell focus/row or cell selection right side. |

| **Ctrl + Up Arrow** | Collapses all tasks. |

| **Ctrl + Down Arrow** | Expands all tasks. |

| **Ctrl + Shift + Up Arrow** | Collapses the selected row. |

| **Ctrl + Shift + Down Arrow** | Expands the selected row. |

| **Enter** | Saves request. |

| **Esc** | Cancels request. |

| **Insert** | Adds a new row. |

| **Ctrl + Insert** | Opens addRowDialog. |

| **Ctrl + F2** | Opens editRowDialog. |

| **Delete** | Deletes the selected row. |

| **Shift + F5** | FocusTask |

- | Ctrl + Shift + F | Focus search |
- | Shift + DownArrow | Extends the row/cell selection downwards. |
- | Shift + UpArrow | Extends the row/cell selection upwards. |
- | Shift + LeftArrow | Extends the cell selection to the left side. |
- | Shift + RightArrow | Extends the cell selection to the right side. |
- | Tab / Shift + Tab | To focus the close icon in the message. |
- | Alt + j | Focus Gantt component. |

Ensuring accessibility

The Gantt component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Gantt component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Gantt component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Globalization in Gantt control

Localization

The [Localization](#) library allows you to localize default text content of the Gantt. The Gantt component has static text on some features (like toolbar area text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the Gantt.

- | Locale key words | Text |
- | ----- | ----- |
- | emptyRecord | No records to display |
- | id | ID |
- | name | Name |
- | startDate | Start Date |
- | endDate | End Date |
- | duration | Duration |
- | progress | Progress |
- | dependency | Dependency |
- | notes | Notes |
- | baselineStartDate | Baseline Start Date |
- | baselineEndDate | Baseline End Date |
- | type | Type |

offset	Offset
resourceName	Resources
resourceID	Resource ID
day	day
hour	hour
minute	minute
days	days
hours	hours
minutes	minutes
generalTab	General
customTab	Custom Columns
writeNotes	Write Notes
addDialogTitle	New Task
editDialogTitle	Task Information
add	Add
edit	Edit
update	Update
delete	Delete
cancel	Cancel
search	Search
task	task
tasks	tasks
zoomIn	Zoom in
zoomOut	Zoom out
zoomToFit	Zoom to fit
expandAll	Expand all
collapseAll	Collapse all
nextTimeSpan	Next timespan
prevTimeSpan	Previous timespan
saveButton	Save
taskBeforePredecessor_FS	You moved "{0}" to start before "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform
taskAfterPredecessor_FS	You moved "{0}" away from "{1}" and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

|taskBeforePredecessor_SS | You moved "{0}" to start before "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform |

|taskAfterPredecessor_SS | You moved "{0}" to start after "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform |

|taskBeforePredecessor_FF | You moved "{0}" to finish before "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform |

|taskAfterPredecessor_FF | You moved "{0}" to finish after "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform |

|taskBeforePredecessor_SF | You moved "{0}" away from "{1}" to starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform |

|taskAfterPredecessor_SF | You moved "{0}" to finish after "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform |

|okText | Ok|

|confirmDelete | Are you sure you want to Delete Record?|

|from | From|

|to | To|

|taskLink | Task Link|

|lag | Lag|

|start | Start|

|finish | Finish|

|enterValue | Enter the value|

|taskInformation | Task Information|

|deleteTask | Delete Task|

|deleteDependency | Delete Dependency|

|convert | Convert|

|save | Save|

|above | Above|

|below | Below|

|child | Child|

|milestone | Milestone|

|toTask | To Task|

|toMilestone | To Milestone|

|eventMarkers | Event markers|

|leftTaskLabel | Left task label|

|rightTaskLabel | Right task label|

|timelineCell | Timeline cell|

|confirmPredecessorDelete | Are you sure you want to remove dependency link?taskMode|

|changeScheduleMode | Change Schedule Mode|

|subTasksStartDate | SubTasks Start Date|

|subTasksEndDate | SubTasks End Date|

|scheduleStartDate | Schedule Start Date|

|scheduleEndDate | Schedule End Date|

|auto | Auto|

|manual | Manual|

|zoomToFit | Zoom to fit|

|excelExport | Excel export|

|csvExport | CSV export|

|pdfExport | Pdf export|

|unit | Unit|

|work | Work|

|taskType | Task Type|

|unassignedTask | Unassigned Task|

|group | Group|

Loading translations

To load translation object in an application use [load](#) function of [L10n](#) class.

The below example demonstrates the Gantt in **Deutsch** culture.

CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
    .Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
    "TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
    rogress("Progress").Child("SubTasks").Locale("de-DE").Render()

<script>
    ej.base.L10n.load({
        'de-DE': {
            'gantt': {
                "id": "Ich würde",
                "name": "Name",
                "startDate": "Anfangsdatum",
                "duration": "Dauer",
                "progress": "Fortschritt",
            }
        }
    });
```

```
</script>
```

LOCALE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}
```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in gantt component.

CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
    ).Child("SubTasks").Locale("de-DE").Render()

<script>
    ej.base.L10n.load({
        'de-DE': {
            'gantt': {
                'id': "Ich würde",
                'name': "Name",
                'startDate': "Anfangsdatum",
                'duration': "Dauer",
                'progress': "Fortschritt",
            }
        }
    });

    loadCultureFiles('de-DE');
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json'];
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (files[prop] === 'numberingSystems.json') {
                ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-data/supplemental/' + files[prop], 'GET', false);
            } else {
                ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
        for (var prop = 0; prop < files.length; prop++) {
```

```

        loadCulture(prop);
    }
}
</script>

```

INTERNATIONALIZATION.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}

```

Note: * In the above sample, **Timeline** is formatted by **NumberFormatOptions** and **DateFormatOptions**.

 * By default, [locale](#) value is **en-US**. If you want to change **en-US** culture, then set the [locale](#).

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Gantt component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Urdu, etc.). To enable RTL Gantt, set the [EnableRtl](#) to true.

CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").Toolbar(new List<string>()
{ "CollapseAll", "ExpandAll" }).TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
).Child("SubTasks").Locale("ar-AE").EnableRtl(true).Render()
<script>
    ej.base.L10n.load({
        'ar-AE': {
            "gantt": {
                "emptyRecord": "لا سجلات لعرضها",
                "id": "هوية شخصية",
                "name": "اسم",
                "startDate": "تاريخ البدء",
                "endDate": "تاريخ الانتهاء",
                "duration": "المدة الزمنية",
                "progress": "تقدم",
                "dependency": "الاعتماد",
                "notes": "ملاحظات",
                "baselineStartDate": "تاريخ البدء الأساسي",
                "baselineEndDate": "تاريخ نهاية خط الأساس",
                "taskMode": "وضع المهام",
                "changeScheduleMode": "تغيير وضع الجدول",
                "subTasksStartDate": "تاريخ بدء المهام الفرعية",
                "subTasksEndDate": "تاريخ انتهاء المهام الفرعية",
                "scheduleStartDate": "جدولة تاريخ البدء",
                "scheduleEndDate": "تاريخ انتهاء الجدول الزمني",
                "auto": "تلقائي",
                "manual": "كتيب",
                "type": "اكتب",
            }
        }
    });

```



```

"offset": "عوض",
"resourceName": "مصادر",
"resourceID": "معرف المورد",
"day": "يوم",
"hour": "ساعة",
"minute": "دقيقة",
"days": "أيام",
"hours": "ساعات",
"minutes": "الدقائق",
"generalTab": "جنرال لواء",
"customTab": "أعمدة مخصصة",
"writeNotes": "اكتب ملاحظات",
"addDialogTitle": "مهمة جديدة",
"editDialogTitle": "معلومات المهمة",
"saveButton": "حفظ",
"add": "إضافة",
"edit": "تعديل",
"update": "تحديث",
"delete": "حذف",
"cancel": "إلغاء",
"search": "بحث",
"task": "مهمة",
"tasks": "مهام",
"zoomIn": "تكبير",
"zoomOut": "تصغير",
"zoomToFit": "تكبير لتناسب",
"excelExport": "اكسل التصدير",
"csvExport": "تصدير CSV",
"expandAll": "توسيع الكل",
"collapseAll": "انهيار جميع",
"nextTimeSpan": "الجدول الزمني التالي",
"prevTimeSpan": "الجدول الزمني السابق",
"okText": "حسنًا",
"confirmDelete": "هل أنت متأكد أنك تريد حذف السجل؟",
"from": "من عند",
"to": "إلى",
"taskLink": "رابط المهمة",
"lag": "بطئ",
"start": "بداية",
"finish": "إنهاء",
"enterValue": "أدخل القيمة",
"taskBeforePredecessor_FS": "0' {قمت بنقل قبل انتهاء} ' للبدء قبل انتهاء. سيتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}' واحدًا أدناه للقيام به",
"taskAfterPredecessor_FS": "0' {قمت بنقل عن 1' {ويتم} ' للبدء بعد انتهاء. سيتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه للقيام به",
"taskBeforePredecessor_SS": "0' {قمت بنقل قبل أن يبدأ} ' للبدء قبل أن يبدأ. سيتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا '{1}' واحدًا أدناه للقيام به",
"taskAfterPredecessor_SS": "0' {قمت بنقل بعد بدء 1' {ويتم} ' للبدء بعد بدء. سيتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه للقيام به",
"taskBeforePredecessor_FF": "0' {قمت بنقل للإنتهاء قبل انتهاء} ' للإنتهاء قبل انتهاء. سيتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}' واحدًا أدناه للقيام به",

```

```

        "taskAfterPredecessor_FF": "للإنهاء بعد انتهاء" 0 '{0} قمت بنقل  

        ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'  

        "واحدًا أدناه للقيام به",  

        "taskBeforePredecessor_SF": "للبدء" 1 '{0} {1} بعيدًا عن  

        التشغيل وترتبط المهمتان. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء  

        "واحدًا أدناه للقيام به",  

        "taskAfterPredecessor_SF": "للإنهاء بعد بدء" 1 '{0} قمت بنقل  

        وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه  

        للقيام به",  

        "taskInformation": "معلومات المهمة",  

        "deleteTask": "حذف المهمة",  

        "deleteDependency": "حذف التبعية",  

        "convert": "تحويل",  

        "save": "حفظ",  

        "above": "في الأعلى",  

        "below": "أدناه",  

        "child": "طفل",  

        "milestone": "معلما",  

        "toTask": "لمهمة",  

        "toMilestone": "إلى معلم",  

        "eventMarkers": "علامات الحدث",  

        "leftTaskLabel": "تسمية المهمة اليسرى",  

        "rightTaskLabel": "تسمية المهمة الصحيحة",  

        "timelineCell": "خلية الجدول الزمني",  

        "confirmPredecessorDelete": "هل أنت متأكد أنك تريد إزالة رابط  

        ؟",  

        "unit": "وحدة",  

        "work": "عمل",  

        "taskType": "نوع المهمة",  

        "unassignedTask": "مهمة غير محددة",  

        "group": "مجموعة",  

        "indent": "مسافة بادئة",  

        "outdent": "عفا عليها الزمن",  

        "segments": "شرائح",  

        "splitTask": "تقسيم المهمة",  

        "mergeTask": "مهمة الدمج",  

        "left": "اليسار",  

        "right": "حق"  

    }  

}
});
</script>

```

ENABLERTL.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}

```

See Also

- [Internationalization](#)

- [Localization](#)

Styling

To modify the Gantt Chart appearance, you need to override the default CSS of gantt chart. Find the list of CSS classes and its corresponding section in Gantt Chart. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

|Section | CSS Class | Purpose of Class|

|-----|-----|-----|

|**Root**|e-gantt|This class is in the root element (div) of the gantt chart control.|

|**Header**|e-gridheader|This class is added in the root element of header element. In this class, You can override thin line between header and content of the gantt chart.|

|e-table|This class is added at 'table' of the gantt chart header. This CSS class makes table width as 100 %.|

|e-columnheader|This class is added at 'tr' of the gantt chart header.|

|**Grid Content**|e-gridcontent|This class is added at root of body content. This is to override background color of the body.|

|e-table|This class is added to table of content. This CSS class makes table width as 100 %.|

|e=row|This class is added to rows of gantt chart.|

|e-altrow|This class is added to alternate rows of gantt chart. This is to override alternate row color of the gantt chart.|

|e-rowcell|This class is added to all cells in the gantt chart. This is to override cells appearance and styling.|

|**Chart Content**|e-gantt-chart|This class is added to the chart side of the gantt chart.|

|e-chart-row|This class is added to rows of gantt chart.|

|**Timeline**|e-timeline-header-container|This class is added to timeline of the gantt chart.|

|e-header-cell-label|This class is added to the header cell of the timeline.|

|e-weekend-header-cell|This class is added to the weekend cells.|

|**Taskbar**|e-taskbar-main-container|This class is added to taskbar of the gantt chart.|

|e-gantt-parent-taskbar|This class is added to the parent task bar of the gantt chart.|

|e-gantt-milestone|This class is added to the milestone tasks of the gantt chart.|

|e-gantt-unscheduled-taskbar|This class is added to the unscheduled tasks.|

|e-gantt-manualparenttaskbar|This class is added to the manual scheduled parent taskbar.|

|e-gantt-child-manualtaskbar|This class is added to the manual scheduled child taskbar.|

|e-gantt-unscheduled-manualtask|This class is added to the manual unscheduled tasks.|

|**Splitter**|e-split-bar|This class is added to the gantt chart splitter.|

|e-resize-handler|This class is added to the resize handler of the gantt chart splitter.|

|e-arrow-left| This class is added to the left arrow of the resize handler. |

|e-arrow=right| This class is added to the right arrow of the resize handler. |

|Connector Lines|e-line| This class is added to the connector lines. |

|e-connector-line-right-arrow| This class is added to the right arrow of the connector line. |

|e-connector-line-left-arrow| This class is added to the left arrow of the connector line. |

|Labels|e-task-label| This class is added to the task labels. |

|e-right-label-container| This class is added to the right label. |

|e-left-label-container| This class is added to the left label. |

|Event Markers|e-event-markers| This class is added to the event markers. |

|Baseline|e-baseline-bar| This class is added to the baseline. |

|e-baseline-gantt-milestone-container| This class is added to the baseline of milestone tasks. |

|Tooltip|e-gantt-tooltip| This class is added to the tooltip. |

Grid lines

In the Gantt control, you can show or hide the grid lines in the TreeGrid side and chart side by using the [GridLines](#) property.

The following options are available in the Gantt control for rendering the grid lines:

- **Horizontal:** The horizontal grid lines alone will be visible.
- **Vertical:** The vertical grid lines alone will be visible.
- **Both:** Both the horizontal and vertical grid lines will be visible on the TreeGrid and chart sides.
- **None:** Gridlines will not be visible on TreeGrid and chart sides.

Note: By default, the [GridLines](#) property is set to Horizontal type.

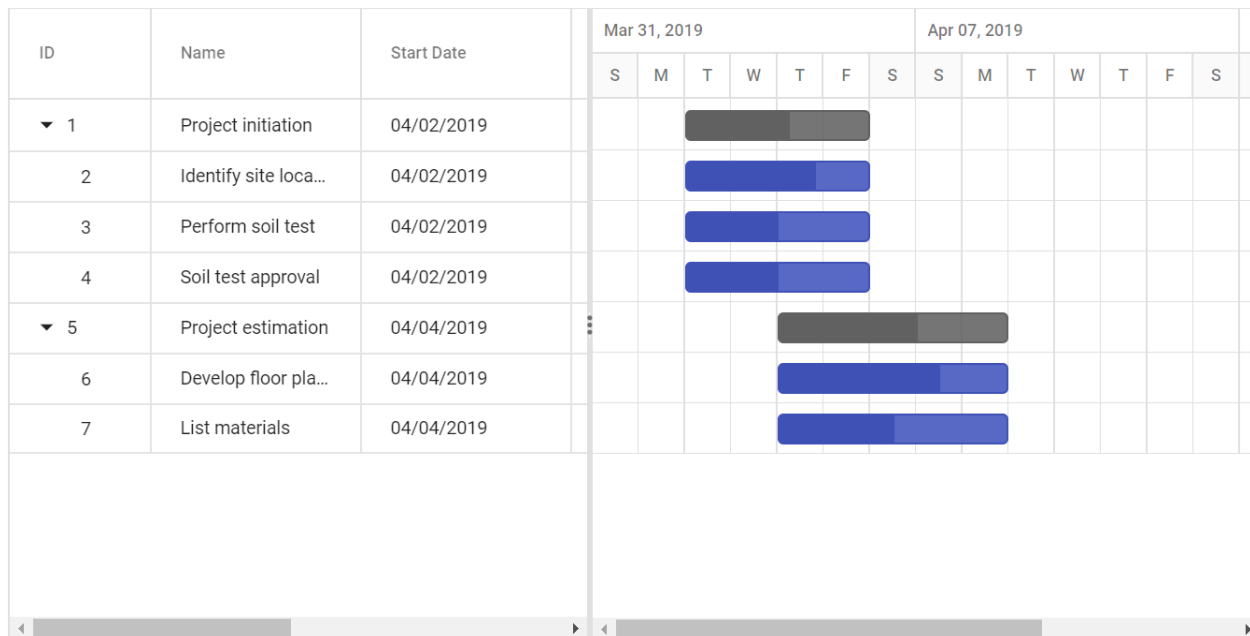
The following code example shows how to change the gridlines rendering mode in the Gantt control.

CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>) ViewBag.DataSource)
.Height("450px").GridLines(
    Syncfusion.EJ2.Gantt.GridLine.Both).TaskFields(ts
=>ts.Id("TaskId").Name("TaskName").StartDate(
"StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Ch
ild("SubTasks")).Render()
```

GRIDLINES.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}
```



Timezone

The Gantt makes use of the current system time zone by default. If it needs to follow some other user-specific time zone, then the `timezone` property needs to be used.

Understanding date manipulation in JavaScript

The `new Date()` in JavaScript returns the exact current date object with complete time and timezone information. For example, it may return value such as `Wed Dec 12 2018 05:23:27 GMT+0530 (India Standard Time)` which indicates that the current date is December 12, 2018 and the current time is 5.23 AM on browsers following the IST timezone.

Display same time everywhere with no time difference

Setting `timezone` to UTC for Gantt will display the same time as in the database for all the users in different time zone.

CSHTML

```
@ (Html.EJS() .Gantt("Gantt") .DataSource((IEnumerable<object>) ViewBag.DataSource)
    .TaskFields(ts =>
ts.Id("TaskId") .Name("TaskName") .StartDate("StartDate") .EndDate("EndDate") .Duration("Duration") .Progress("Progress")
    .Dependency("Predecessor") .ParentID("ParentID"))
    .Height("450px") .IncludeWeekend(true) .DurationUnit(Syncfusion.EJ2.Gantt.DurationUnit.Hour)
    .TimelineSettings(ts =>
ts.TimelineUnitSize(65) .TopTier(tt => tt.Format("MMM dd, yyyy") .Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Day))
    .BottomTier(bt => bt.Format("hh:mm a") .Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Hour))
    .Timezone("UTC") .DateFormat("hh:mm a")
    .DayWorkingTime(dt =>
```

```
        {  
            dt.From(0).To(24).Add();  
        })  
        .Render()  
    )
```

SAME-TIME.CS

```
public IActionResult Index()  
{  
    ViewBag.DataSource = ganttData();  
    return View();  
}  
  
public static List<GanttDataSource> ganttData()  
{  
    List<GanttDataSource> GanttDataSourceCollection = new  
List<GanttDataSource>();  
    GanttDataSource Record1 = new GanttDataSource()  
    {  
        TaskId = 1,  
        TaskName = "Project schedule",  
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),  
        EndDate = new DateTime(2019, 03, 10)  
    };  
    GanttDataSource Record2 = new GanttDataSource()  
    {  
        TaskId = 2,  
        TaskName = "Planning",  
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),  
        EndDate = new DateTime(2019, 02, 10),  
        ParentID = 1  
    };  
    GanttDataSource Record3 = new GanttDataSource()  
    {  
        TaskId = 3,  
        TaskName = "Plan timeline",  
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),  
        EndDate = new DateTime(2019, 02, 10),  
        Duration = 6,  
        Progress = 60,  
        ParentID = 2  
    };  
    GanttDataSource Record4 = new GanttDataSource()  
    {  
        TaskId = 4,  
        TaskName = "Plan budget",  
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),  
        EndDate = new DateTime(2019, 02, 10),  
        Duration = 6,  
        Progress = 90,  
        ParentID = 2  
    };  
    GanttDataSource Record5 = new GanttDataSource()  
    {  
        TaskId = 5,  
        TaskName = "Allocate resources",
```

```
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        Duration = 6,
        Progress = 75,
        ParentID = 2
    };
    GanttDataSource Record6 = new GanttDataSource()
    {
        TaskId = 6,
        TaskName = "Planning complete",
        StartDate = new DateTime(2019, 02, 06, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        Duration = 0,
        Predecessor = "3, 4, 5",
        ParentID = 2
    };
    GanttDataSource Record7 = new GanttDataSource()
    {
        TaskId = 7,
        TaskName = "Design",
        StartDate = new DateTime(2019, 02, 13, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 17),
        ParentID = 1
    };
    GanttDataSource Record8 = new GanttDataSource()
    {
        TaskId = 8,
        TaskName = "Software specification",
        StartDate = new DateTime(2019, 02, 13, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 15),
        Duration = 3,
        Progress = 60,
        Predecessor = "6",
        ParentID = 7
    };
    GanttDataSource Record9 = new GanttDataSource()
    {
        TaskId = 9,
        TaskName = "Develop prototype",
        StartDate = new DateTime(2019, 02, 13, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 15),
        Duration = 3,
        Progress = 100,
        Predecessor = "6",
        ParentID = 7
    };
    GanttDataSource Record10 = new GanttDataSource()
    {
        TaskId = 10,
        TaskName = "Get approval from customer",
        StartDate = new DateTime(2019, 02, 16, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 17),
        Duration = 2,
        Progress = 100,
        Predecessor = "9",
        ParentID = 7
    };
};
```

```

GanttDataSource Record11 = new GanttDataSource()
{
    TaskId = 11,
    TaskName = "Design complete",
    StartDate = new DateTime(2019, 02, 17, 08, 00, 00),
    EndDate = new DateTime(2019, 02, 17),
    Duration = 0,
    Predecessor = "10",
    ParentID = 7
};
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
GanttDataSourceCollection.Add(Record3);
GanttDataSourceCollection.Add(Record4);
GanttDataSourceCollection.Add(Record5);
GanttDataSourceCollection.Add(Record6);
GanttDataSourceCollection.Add(Record7);
GanttDataSourceCollection.Add(Record8);
GanttDataSourceCollection.Add(Record9);
GanttDataSourceCollection.Add(Record10);
GanttDataSourceCollection.Add(Record11);
return GanttDataSourceCollection;
}
public class GanttDataSource
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public DateTime? StartDate { get; set; }
    public DateTime? EndDate { get; set; }
    public int? Duration { get; set; }
    public int Progress { get; set; }
    public string Predecessor { get; set; }
    public int ParentID { get; set; }
}

```

CRUD operations with timezone

CRUD operations can be performed with timezone, and the gantt is rendered based on the timezone specified in the load time. All the editing actions will be done based on the user timezone, but on database save action, we have reversed this conversion to local time and provided data to client side events for better understanding purpose. Refer to the following code example.

CSHTML

```

@(Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
    .TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
        .Dependency("Predecessor").ParentID("ParentID"))
    .Height("450px").IncludeWeekend(true).DurationUnit(Syncfusion.EJ2.Gantt.DurationUnit.Hour)
    .EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true).AllowTaskbarEditing(true).ShowDeleteConfirmDialog(true))

```



```

        .TimelineSettings(ts =>
ts.TimelineUnitSize(65).TopTier(tt => tt.Format("MMM dd,
yyyy").Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Day))
        .BottomTier(bt => bt.Format("hh:mm
a").Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Hour))
        .Timezone("America/New_York").DateFormat("hh:mm a")
        .DayWorkingTime(dt =>
        {
            dt.From(0).To(24).Add();
        })
        .Render()
    )

```

TIMEZONE-CRUD.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = ganttData();
    return View();
}

public static List<GanttDataSource> ganttData()
{
    List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
    GanttDataSource Record1 = new GanttDataSource()
    {
        TaskId = 1,
        TaskName = "Project schedule",
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),
        EndDate = new DateTime(2019, 03, 10)
    };
    GanttDataSource Record2 = new GanttDataSource()
    {
        TaskId = 2,
        TaskName = "Planning",
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        ParentID = 1
    };
    GanttDataSource Record3 = new GanttDataSource()
    {
        TaskId = 3,
        TaskName = "Plan timeline",
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        Duration = 6,
        Progress = 60,
        ParentID = 2
    };
    GanttDataSource Record4 = new GanttDataSource()
    {
        TaskId = 4,
        TaskName = "Plan budget",
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        Duration = 6,
    }
}

```

```
        Progress = 90,
        ParentID = 2
    };
    GanttDataSource Record5 = new GanttDataSource()
    {
        TaskId = 5,
        TaskName = "Allocate resources",
        StartDate = new DateTime(2019, 02, 04, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        Duration = 6,
        Progress = 75,
        ParentID = 2
    };
    GanttDataSource Record6 = new GanttDataSource()
    {
        TaskId = 6,
        TaskName = "Planning complete",
        StartDate = new DateTime(2019, 02, 06, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 10),
        Duration = 0,
        Predecessor = "3, 4, 5",
        ParentID = 2
    };
    GanttDataSource Record7 = new GanttDataSource()
    {
        TaskId = 7,
        TaskName = "Design",
        StartDate = new DateTime(2019, 02, 13, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 17),
        ParentID = 1
    };
    GanttDataSource Record8 = new GanttDataSource()
    {
        TaskId = 8,
        TaskName = "Software specification",
        StartDate = new DateTime(2019, 02, 13, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 15),
        Duration = 3,
        Progress = 60,
        Predecessor = "6",
        ParentID = 7
    };
    GanttDataSource Record9 = new GanttDataSource()
    {
        TaskId = 9,
        TaskName = "Develop prototype",
        StartDate = new DateTime(2019, 02, 13, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 15),
        Duration = 3,
        Progress = 100,
        Predecessor = "6",
        ParentID = 7
    };
    GanttDataSource Record10 = new GanttDataSource()
    {
        TaskId = 10,
        TaskName = "Get approval from customer",
```

```

        StartDate = new DateTime(2019, 02, 16, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 17),
        Duration = 2,
        Progress = 100,
        Predecessor = "9",
        ParentID = 7
    };
    GanttDataSource Record11 = new GanttDataSource()
    {
        TaskId = 11,
        TaskName = "Design complete",
        StartDate = new DateTime(2019, 02, 17, 08, 00, 00),
        EndDate = new DateTime(2019, 02, 17),
        Duration = 0,
        Predecessor = "10",
        ParentID = 7
    };
    GanttDataSourceCollection.Add(Record1);
    GanttDataSourceCollection.Add(Record2);
    GanttDataSourceCollection.Add(Record3);
    GanttDataSourceCollection.Add(Record4);
    GanttDataSourceCollection.Add(Record5);
    GanttDataSourceCollection.Add(Record6);
    GanttDataSourceCollection.Add(Record7);
    GanttDataSourceCollection.Add(Record8);
    GanttDataSourceCollection.Add(Record9);
    GanttDataSourceCollection.Add(Record10);
    GanttDataSourceCollection.Add(Record11);
    return GanttDataSourceCollection;
}

public class GanttDataSource
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public DateTime? StartDate { get; set; }
    public DateTime? EndDate { get; set; }
    public int? Duration { get; set; }
    public int Progress { get; set; }
    public string Predecessor { get; set; }
    public int ParentID { get; set; }
}

```

Timezone methods

offset

This method is used to calculate the difference between passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
Timezone	String	Timezone.

Returns **number**

`sh

```
// Assume your local timezone as IST/UTC+05:30
var timezone = new ej.schedule.Timezone();
var date = new Date(2018,11,5,15,25,11);
var timeZoneOffset = timezone.offset(date,"Europe/Paris");
console.log(timeZoneOffset); //-60
`
```

convert

This method is used to convert the passed date from one timezone to another timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
fromOffset	number/string	Timezone from which date need to be converted.
toOffset	number/string	Timezone to which date need to be converted.

Returns **Date**

```
`sh
// Assume your local timezone as IST/UTC+05:30
var timezone = new ej.schedule.Timezone();
var date = new Date(2018,11,5,15,25,11);
var convertedDate = timezone.convert(date, "Europe/Paris", "Asia/Tokyo");
var convertedDate1 = timezone.convert(date, 60, -360);
console.log(convertedDate); //2018-12-05T17:55:11.000Z
console.log(convertedDate1); //2018-12-05T16:55:11.000Z
`
```

remove

This method is used to remove the time difference between passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC as date object.
Timezone	String	Timezone.

Returns **Date**

```
`sh
// Assume your local timezone as IST/UTC+05:30
var timezone = new ej.schedule.Timezone();
var date = new Date(2018,11,5,15,25,11);
```

```
var convertedDate = timezone.remove(date, "Europe/Paris");
console.log(convertedDate); //2018-12-05T14:25:11.000Z
,
```

Migration from Essential JS 1

This topic shows the API equivalent of JS2 Gantt component to be used, while migrating your project that uses JS1 Gantt.

Data Binding and Task mapping

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Data Binding | **Property:** *Datasource*

@(Html.EJ().Gantt("Gantt")
.Datasource(ViewBag.datasources)
) | **Property:** *Datasource*

@Html.EJS().Gantt("Gantt")
.DataSource((IEnumerable<object>)ViewBag.DataSources)
.Render() |

| To map id of task from data source | **Property:** *TaskIdMapping*

@(Html.EJ().Gantt("Gantt")
.TaskIdMapping("Id")
) | **Property:** *TaskFields.Id*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.Id("TaskId")).Render() |

| To map name of task from data source | **Property:** *TaskNameMapping*

@(Html.EJ().Gantt("Gantt")
.TaskNameMapping("Id")
) | **Property:** *TaskFields.Name*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.Name("TaskName")).Render() |

| To map start date from data source | **Property:** *StartDateMapping*

@(Html.EJ().Gantt("Gantt")
.StartDateMapping("StartDate")
) | **Property:** *TaskFields.StartDate*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.StartDate("StartDate")).Render() |

| To map end date from data source | **Property:** *EndDateMapping*

@(Html.EJ().Gantt("Gantt")
.EndDateMapping("EndDate")
) | **Property:** *TaskFields.EndDate*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.EndDate("EndDate")).Render() |

| To map duration from data source | **Property:** *DurationMapping*

@(Html.EJ().Gantt("Gantt")
.DurationMapping("Duration")
) | **Property:** *TaskFields.Duration*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.Duration("Duration")).Render() |

| To map duration unit from data source | **Property:** *DurationUnitMapping*

@(Html.EJ().Gantt("Gantt")
.DurationUnitMapping("DurationUnit")
) | **Property:** *TaskFields.DurationUnit*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.DurationUnit("DurationUnit")).Render() |

| To map predecessors from data source | **Property:** *PredecessorMapping*

@(Html.EJ().Gantt("Gantt")
.PredecessorMapping("Predecessor")
) | **Property:** *TaskFields.Dependency*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.Dependency("Predecessor")).Render() |

| To map progress from data source | **Property:** *ProgressMapping*

 @(Html.EJ().Gantt("Gantt")
.ProgressMapping("PercentDone")
) | **Property:**
TaskFields.Progress

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.Progress("Progress")).Render() |

| To map child task from data source | **Property:** *ChildMapping*

 @(Html.EJ().Gantt("Gantt")
.ChildMapping("Children")
) | **Property:** *TaskFields.child*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.Child("SubTasks")).Render() |

| To map baseline start date from data source | **Property:** *BaselineStartDateMapping*

 @(Html.EJ().Gantt("Gantt")
.BaselineStartDateMapping("baselineStartDate")
) |
Property: *TaskFields.BaselineStartDate*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.BaselineStartDate("BaselineStartDate")).Render() |

| To map baseline end date from data source | **Property:** *BaselineEndDateMapping*

 @(Html.EJ().Gantt("Gantt")
.BaselineEndDateMapping("baselineEndDate")
) | **Property:**
TaskFields.BaselineEndDate

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.BaselineEndDate("BaselineEndDate")).Render() |

| To map milestone mapping from data source | **Property:** *MilestoneMapping*

 @(Html.EJ().Gantt("Gantt")
.MilestoneMapping("isMileStone")
) | **Property:**
TaskFields.Milestone

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.Milestone("isMileStone")).Render() |

| To map notes from data source | **Property:** *NotesMapping*

 @(Html.EJ().Gantt("Gantt")
.NotesMapping("notesContent")
) | **Property:** *TaskFields.Notes*

@Html.EJS().Gantt("Gantt")
.TaskFields(ts => ts.Notes("Notes")).Render() |

| To map parent task id from data source | **Property:** *ParentTaskIdMapping*

 @(Html.EJ().Gantt("Gantt")
.ParentTaskIdMapping("ParentId")
) | **Property:**
TaskFields.ParentId

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.ParentId("ParentId")).Render() |

| To map assigned resources from data source | **Property:** *ResourceInfoMapping*

 @(Html.EJ().Gantt("Gantt")
.ResourceInfoMapping("Resources")
) | **Property:**
TaskFields.ResourceInfo

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.ResourceInfo("ResourceId")).Render() |

| To map expand state from data source | **Property:** *ExpandStateMapping*

 @(Html.EJ().Gantt("Gantt")
.ExpandStateMapping("ExpandState")
) | **Property:**
TaskFields.ExpandState

@Html.EJS().Gantt("Gantt")
.TaskFields(ts =>
 ts.ExpandState("isExpand")).Render() |

Sorting

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| Default | **Property:** *AllowSorting*

 @(Html.EJ().Gantt("Gantt")
.AllowSorting(true)
) | **Property:** *AllowSorting*

@Html.EJS().Gantt("Gantt")
.AllowSorting(true).Render() |

| To enable/disable multiple sorting option | **Property:** *AllowMultiSorting*

@(Html.EJ().Gantt("Gantt")
.AllowMultiSorting(true)
) | **Property:** *AllowSorting*

@Html.EJS().Gantt("Gantt")
.AllowSorting(true).Render() |

| Sort column Initially | **Property:** *SortSettings.SortedColumns*

@(Html.EJ().Gantt("Gantt")
.AllowSorting(true)
.SortSettings(sort=>{
sort.SortedColumns(sorted => sorted.Field("taskName").Direction(SortOrder.Descending).Add());
})
) |

Property: *SortSettings.Columns*

@Html.EJS().Gantt("Gantt")
.AllowSorting(true)
.SortSettings(ss=>
 ss.Columns(col=>
{
col.Field("TaskName")
.Direction(Syncfusion.EJ2.Gantt.SortDirection.Ascending).Add();

})).Render() |

| Clear the Sorted columns | **Method:** *clearSorting()*

@(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
\$("#Gantt").data("ejGantt");
ganttObj.clearSorting();
</script> | **Method:** *clearSorting()*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
ganttObj.clearSorting();
</script>
|

| Sort records in Gantt | **Method:** *sortColumn(mappingName, columnSortDirection)*

@ (Html.EJ()).Gantt("Gantt")

Script:
<script>
var ganttObj =
\$("#Gantt").data("ejGantt");
ganttObj.sortColumn("startDate", "ascending");
</script> |

Method: *sortColumn(columnName, direction,[isMultiSort])*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
ganttObj.sortColumn('startDate','ascending');
</script> |

Filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Filter column Initially | **Property:** *FilterSettings.FilteredColumns*

@(Html.EJ().Gantt("Gantt")
.FilterSettings(filter =>
{
 filter.FilteredColumns(filtered =>
{
 filtered.Value("plan").Field("taskName").Predicate("and")
 .Operator(FilterOperatorType.StartsWith).Add();
 });
 })
}) | **Property:**
FilterSettings.Columns

@{
List<object> filterColumns = new
List<object>();
filterColumns.Add(new { field = "TaskName", matchCase = false, @operator =
"startswith", predicate = "and", value = "Identify"
});
}

@Html.EJS().Gantt("Gantt")
.AllowFiltering(true)
.FilterSettings(filter =>
filter.Columns(filterColumns)).Render() |

| Filter records in Gantt | **Method:** *filterColumn(fieldName, filterOperator, filterValue, [predicate],*

[matchCase]

@ (Html.EJ()).Gantt("Gantt")

Script:
<script>
var ganttObj
= \$("#Gantt").data("ejGantt");
ganttObj.filterColumn("taskName", "startswith",
"plan");
</script> | **Method:** *filterByColumn(fieldName, filterOperator, filterValue, [predicate],*
[matchCase],[ignoreAccent])

```
</></>@Html.EJS().Gantt("Gantt").Render()<br><br>Script:<br><script><br>var ganttObj =
document.getElementById('Gantt').ej2_instances[0];<br>ganttObj.filterByColumn('taskName',
'startswith', 'plan');<br></script> |
```

| Filter multiple columns | **Method:** *filterContent(ejPredicate)*

 @Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
var predicate = ej.Predicate("taskName",
 ej.FilterOperators.equal, "planning", false)
 .or("taskName",
 ej.FilterOperators.equal, "plan budget", false)
 .and("progress",
 ej.FilterOperators.equal, 100, true);
ganttObj.filterContent(ejPredicate);
</script> | Not
 applicable |

| Clear filtered columns | **Method:** *clearFilter()*

@Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.clearFilter();
</script> | **Method:** *clearFiltering()*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.clearFiltering();
</script>
 |

Searching

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *ToolbarSettings.ToolbarItems*

 @Html.EJ().Gantt("Gantt")
.ToolbarSettings(toolbar=>
 {
 toolbar.ShowToolbar(true);
 toolbar.ToolbarItems(new
 List<GanttToolBarItems>() { GanttToolBarItems.Search });
})
 | **Property:** *Toolbar*

 @Html.EJS().Gantt("Gantt")
.Toolbar(new List<string>() { "Search" }).Render()
 |

| Search records in Gantt | **Method:** *searchItem(key)*

@Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.searchItem("plan");
</script> | **Method:**
search(key)

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var
 ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.search('plan');
</script> |

Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowSelection*

 @Html.EJ().Gantt("Gantt")
.AllowSelection(true)
 | **Property:** *AllowSelection*

@Html.EJS().Gantt("Gantt")
.AllowSelection(true).Render() |

| To define selection type in Gantt | **Property:** *SelectionType*

 @Html.EJ().Gantt("Gantt")
.SelectionType(GanttSelectionType.Multiple)
 | **Property:**
SelectionSettings.Type

@Html.EJS().Gantt("Gantt")
.SelectionSettings(ss =>
 ss.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).Render() |

| To define selection mode in Gantt | **Property:** *SelectionMode*

 @(Html.EJ().Gantt("Gantt")
.SelectionMode(GanttSelectionMode.Row)
)| **Property:**
SelectionSettings.Mode

@Html.EJS().Gantt("Gantt")
.SelectionSettings(ss=>
ss.Mode(Syncfusion.EJ2.G
 rids.SelectionMode.Both)).Render() |

| Select Row by Index | **Property:** *SelectedRowIndex*

 @(Html.EJ().Gantt("Gantt")
.SelectedRowIndex(3)
)| **Property:** *SelectedRowIndex*

@Html.EJS().Gantt("Gantt")
.SelectedRowIndex(3).Render() |

| To define selected cell index in Gantt | **Property:** *SelectedCellIndexes*

 @(Html.EJ().Gantt("Gantt")
.SelectedCellIndexes([])
)| Not applicable |

| Select Multiple Cells | **Method:** *selectCells(Indexes,preservePreviousSelectedCell)*

 @(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
var indexes = [{rowIndex:4, cellIndex: 4}, {rowIndex: 3,
 cellIndex: 3}];
ganttObj.selectCells(indexes, true);
</script>| Not Applicable |

| Select multiple Rows | **Method:** *selectMultipleRows(rowIndexes)*

@(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.selectMultipleRows([1,2,3]);
</script> | **Method:**
selectRows(key)

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.selectionModule.selectRows(
 [1,2,3]);
</script> |

| Triggers after cell selection action | **Event:** *CellSelected*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.CellSelected("cellSelected"))

Script:
<script>
function cellSelected() {
 }
</script> | **Event:** *CellSelected*

@Html.EJS().Gantt("Gantt")
.CellSelected("cellSelected").Render()

Script:<
 br><script>
function cellSelected() { }
</script> |

| Triggers on cell selection action | **Event:** *CellSelecting*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.CellSelecting("cellSelecting"))

Script:
<script>
function cellSelecting() {
 }
</script> | **Event:** *CellSelecting*

@Html.EJS().Gantt("Gantt")
.CellSelecting("cellSelecting").Render()

Script:

<script>
function cellSelecting() { }
</script> |

| Triggers after row selection action | **Event:** *RowSelected*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.RowSelected("rowSelected"))

Script:
<script>
function rowSelected() {
 }
</script> | **Event:** *RowSelected*

@Html.EJS().Gantt("Gantt")
.RowSelected("rowSelected").Render()

Script:

<script>
function rowSelected() { }
</script> |

| Triggers before row selection action | **Event:** *RowSelecting*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e

```
=><br>e.RowSelecting("rowSelecting"))<br><br>Script:<br><script><br>function rowSelecting() {
}<br><script> | Event: RowSelecting
<br><br>@Html.EJS().Gantt("Gantt")<br>.RowSelecting("rowSelecting").Render()<br><br>Script
:<br><script><br>function rowSelecting() { }<br><script> |
```

Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| Default | **Property:** *EditSettings*

 @Html.EJ().Gantt("Gantt")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.AllowAdding(true);
edit.AllowDeleting(true);
edit.S
howDeleteConfirmDialog(true);
})
 | **Property:** *EditSettings*

@Html.EJS().Gantt("Gantt")
.EditSettings(es=>es.AllowEditing(true)
.AllowAdding(true)<
br>.AllowDeleting(true)
.ShowDeleteConfirmDialog(true)).Render() |

| Cell Editing | **Property:** *EditSettings.editMode*

@Html.EJ().Gantt("Gantt")
.EditSettings(edit=>
{
edit.EditMode("cellEditing");
})
 | **Property:** *EditSettings.mode*

@Html.EJS().Gantt("Gantt")
.EditSettings(es=>
es.Mode(Syncfusion.EJ2.Gantt.EditMode.
Auto)).Render() |

| Dialog Editing | **Property:** *EditSettings.editMode*

@Html.EJ().Gantt("Gantt")
.EditSettings(edit=>
{
edit.EditMode("normal");
})
 | **Property:** *EditSettings.mode*

@Html.EJS().Gantt("Gantt")
.EditSettings(es=>
es.Mode(Syncfusion.EJ2.Gantt.EditMode.
Dialog)).Render() |

| To enable/disable taskbar editing | **Property:** *AllowGanttChartEditing*

@Html.EJ().Gantt("Gantt")
.AllowGanttChartEditing(true)
 | **Property:**
EditSettings.allowTaskbarEditing
@Html.EJS().Gantt("Gantt")
.EditSettings(es
=>
es.AllowTaskbarEditing(true)).Render() |

| To enable progressbar resizing | **Property:** *EnableProgressBarResizing*

@Html.EJ().Gantt("Gantt")
.EnableProgressBarResizing(false)
 | **Property:**
EditSettings.allowTaskbarEditing

 @Html.EJS().Gantt("Gantt")
.EditSettings(es
=>
es.AllowTaskbarEditing(true)).Render() |

| To enable indent/ outdent option | **Property:** *EditSettings.allowIndent*

@Html.EJ().Gantt("Gantt")
.EditSettings(edit =>
{
edit.AllowIndent(true);
})
 | Not applicable |

| To define click or double click action to begin edit action | **Property:** *EditSettings.beginEditAction*

@Html.EJ().Gantt("Gantt")
.EditSettings(edit =>
{
edit.BeginEditAction(GanttBeginEditAction.Click);
})
 | Not applicable |

| To define new row position in Gantt | **Property:** *EditSettings.rowPosition*

@Html.EJ().Gantt("Gantt")
.EditSettings(edit=>
{
edit.RowPosition(GanttRowPosition.Child);
})
 | **Property:**
EditSettings.newRowPosition


```
@Html.EJS().Gantt("Gantt")<br>.EditSettings(es=><br>es.NewRowPosition(Syncfusion.EJ2.Gantt
.RowPosition.Child))<br>.Render() |
```

| To define fields in edit dialog | **Property:** *EditDialogFields*


```
@(Html.EJ().Gantt("Gantt")<br>.EditDialogFields(editDialog =>
{<br>&#160;editDialog.Field("TaskID").EditType("stringedit").Add();<br>&#160;editDialog.Field("
TaskName").EditType("stringedit").Add();<br>})<br>) | Property: EditDialogFields <br/><br/>
@Html.EJS().Gantt("Gantt")<br>.EditDialogFields(edf =>>
{<br>&#160;edf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.General)<br>&#160;.HeaderText("Ge
neral").Add();<br>&#160;edf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Dependency)<br>.Add()
;<br>}).Render() |
```

| To define fields in add dialog | **Property:** *AddDialogFields*


```
@(Html.EJ().Gantt("Gantt")<br>.AddDialogFields(addDialog =>
{<br>&#160;addDialog.Field("TaskID").EditType("stringedit").Add();<br>&#160;addDialog.Field("T
askName").EditType("stringedit").Add();<br>})<br>) | Property: AddDialogFields <br/><br/>
@Html.EJS().Gantt("Gantt")<br>.AddDialogFields(adf=>
{<br>&#160;adf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.General)<br>.HeaderText("General
Tab").Add();<br>&#160;adf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Dependency)<br>.Add();<br>
}).Render() |
```

| To make Gantt as read only | **Property:** *ReadOnly*


```
@(Html.EJ().Gantt("Gantt")<br>.ReadOnly(true)<br>)| Not Applicable |
```

| To open Edit dialog | **Method:** *openEditDialog()*

```
<br/><br/>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>ganttObj.openEditDialog();<br></script> | Method:
openEditDialog()
<br/><br/>@Html.EJS().Gantt("Gantt").Render()<br><br>Script:<br><script><br>var ganttObj =
document.getElementById('Gantt').ej2_instances[0];<br>ganttObj.openEditDialog();<br></scrip
t> |
```

| To open Add dialog | **Method:** *openAddDialog()*

```
<br/><br/>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>ganttObj.openAddDialog();<br></script> | Method:
openAddDialog()
<br/><br/>@Html.EJS().Gantt("Gantt").Render()<br><br>Script:<br><script><br>var ganttObj =
document.getElementById('Gantt').ej2_instances[0];<br>ganttObj.openAddDialog();<br></scrip
t> |
```

| Add task in Gantt | **Method:** *addRecord(data, rowPosition)*

```
<br/><br/>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>var data = {<br>&#160;&#160;taskId:
40,<br>&#160;&#160;taskName:"New Task
40",<br>&#160;&#160;startDate:"2/20/2014",<br>&#160;&#160;endDate:"2/25/2014"<br>};<br>
>ganttObj.addRecord(data, ej.Gantt.AddRowPosition.Child);<br></script> | Method:
addRecord(data, rowPosition, rowIndex)
<br/><br/>@Html.EJS().Gantt("Gantt").Render()<br><br>Script:<br><script><br>var data =
```

```
{<br/>&#160;&#160;taskId:"40",<br/>&#160;&#160;taskName:"New Task
40",<br/>&#160;&#160;startDate:"2/20/2014",<br/>&#160;&#160;endDate:"2/25/2014"<br>};<br/
>gantObj.addRecord(data, 'Below', 10);<br/></script> |
```

| Delete selected item | **Method:** *deleteItem()*

```
<br/><br/>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>gantObj.deleteItem();<br/></script> | Method: deleteRecord()
<br/><br/>@(Html.EJS().Gantt("Gantt").Render())<br><br>Script:<br><script><br>var ganttObj =
document.getElementById('Gantt').ej2_instances[0];<br>gantObj.editModule.deleteRecord();<
br/></script> |
```

| Update task details by id | **Method:** *updateRecordByTaskId(data)*

```
<br/><br/>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>var data = { taskId: 4, taskName: "updated
value"};<br>gantObj.updateRecordByTaskId(data);<br/></script> | Method: updateRecordById
<br/><br/>@(Html.EJS().Gantt("Gantt").Render())<br><br>Script:<br><script><br>var data = {
taskId: 4, taskName: "updated value"};<br>gantObj.updateRecordById(data);<br/></script> |
```

| Delete dependency | **Method:** *deleteDependency(fromTaskId,toTaskId)*

```
<br/><br/>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>gantObj.deleteDependency(3, 6);<br/></script> | Not
applicable |
```

| Save Edit | **Method:** *saveEdit()*


```
@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>gantObj.saveEdit();<br/></script> | Not applicable |
```

| Cancel Edit | **Method:** *cancelEdit()*


```
@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>cancelEdit var ganttObj =
$("#Gantt").data("ejGantt");<br>gantObj.cancelEdit();<br/></script> | Method: cancelEdit()
<br><br>@(Html.EJ().Gantt("Gantt"))<br><br>Script:<br><script><br>var ganttObj =
$("#Gantt").data("ejGantt");<br>gantObj.cancelEdit();<br/></script> |
```

| Triggers for every Gantt action before it get started | **Event:** *ActionBegin*

```
<br/><br/>@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
=><br>e.ActionBegin("actionBegin"))<br><br>Script:<br><script><br>function actionBegin() {
}<br></script> | Event: ActionBegin
<br/><br/>@(Html.EJS().Gantt("Gantt")<br>.ActionBegin("actionBegin").Render())<br><br>Script:<
br><script><br>function actionBegin() { }<br></script> |
```

| Triggers for after every Gantt action completed | **Event:** *ActionComplete*

```
<br/><br/>@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
=><br>e.ActionComplete("actionComplete"))<br><br>Script:<br><script><br>function
actionComplete() { }<br></script> | Event: ActionComplete
<br/><br/>@(Html.EJS().Gantt("Gantt")<br>.ActionComplete("actionComplete").Render())<br><br>
Script:<br><script><br>function actionComplete() { }<br></script> |
```

| Triggers while resizing, dragging the taskbar | **Event:** *TaskbarEditing*

```
<br/><br/>@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
```

```

=><br>e.TaskbarEditing("taskbarEditing"))<br><br>Script:<br><script><br>function
taskbarEditing() { }<br><script> | Event: TaskbarEditing
<br><br>@Html.EJS().Gantt("Gantt")<br>.TaskbarEditing("taskbarEditing").Render()<br><br>Script:<br><script><br>function taskbarEditing() { }<br><script> |
| Triggers after taskbar resize, drag action | Event: TaskbarEdited
<br><br>@Html.EJS().Gantt("Gantt").ClientSideEvents(e
=><br>e.TaskbarEdited("taskbarEdited"))<br><br>Script:<br><script><br>function taskbarEdited()
{ }<br><script> | Event: TaskbarEdited
<br><br>@Html.EJS().Gantt("Gantt")<br>.TaskbarEdited("taskbarEdited").Render()<br><br>Script:<br><script><br>function taskbarEdited() { }<br><script> |

```

Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To enable/disable column resize | **Property:** AllowColumnResize

@Html.EJS().Gantt("Gantt")
.AllowColumnResize(true)
)| **Property:** AllowResizing

 @Html.EJS().Gantt("Gantt")
.AllowResizing(true).Render()|

| To enable/disable column chooser | **Property:** ShowColumnChooser

@Html.EJS().Gantt("Gantt")
.ShowColumnChooser(true)
)| **Property:** ShowColumnMenu

@Html.EJS().Gantt("Gantt")
.ShowColumnMenu(true).Render()|

| To enable/disable column add, remove option in column menu | **Property:** ShowColumnOptions

 @Html.EJS().Gantt("Gantt")
.ShowColumnOptions(true)
)| Not applicable|

| Tree column index | **Property:** TreeColumnIndex

@Html.EJS().Gantt("Gantt")
.TreeColumnIndex(2)
)| **Property:** TreeColumnIndex

@Html.EJS().Gantt("Gantt")
.TreeColumnIndex(2).Render()|

| To define column fields in column menu | **Property:** ColumnDialogFields

@Html.EJS().Gantt("Gantt")
.ColumnDialogFields(new List <GanttColumnDialogFields>()
{
 GanttColumnDialogFields.Field,
 GanttColumnDialogFields.HeaderText,

> GanttColumnDialogFields.EditType
}) | Not applicable |

| Show column | **Method:** showColumn(headerText)

@Html.EJS().Gantt("Gantt")

Script:
<script>
var ganttObj =
\$("#Gantt").data("ejGantt");
ganttObj.showColumn("Task Name");
</script> | **Method:**
showColumn(keys, showBy)

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
ganttObj.showColumn("TaskName");

</script>|

| Hide column | **Method:** hideColumn(headerText)

@Html.EJS().Gantt("Gantt")

Script:
<script>
var ganttObj =
\$("#Gantt").data("ejGantt");
ganttObj.hideColumn("Task Name");
</script> | **Method:**
hideColumn(keys, showBy)

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =

```
document.getElementById('Gantt').ej2_instances[0];<br>ganttObj.hideColumn("TaskName");<br></script>
```

Toolbar

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To configure default toolbars of Gantt | **Property:** *ToolbarSettings.toolbarItems*


```
@(Html.EJ().Gantt("Gantt")<br>.ToolbarSettings(toolbar =>
{<br>&#160;toolbar.ShowToolbar(true);<br>&#160;toolbar.ToolbarItems(new
List<GanttToolBarItems>()
{<br>&#160;&#160;GanttToolBarItems.Add,<br>&#160;&#160;GanttToolBarItems.Edit,<br>&#160;
&#160;GanttToolBarItems.Delete,<br>&#160;&#160;GanttToolBarItems.Update,<br>&#160;&#160;
GanttToolBarItems.Cancel,<br>&#160;&#160;GanttToolBarItems.ExpandAll,<br>&#160;&#160;GanttToolBarItems.CollapseAll,<br>&#160;&#160;GanttToolBarItems.Search,<br>&#160;&#160;GanttToolBarItems.PrevTimeSpan,<br>&#160;&#160;GanttToolBarItems.NextTimeSpan<br>&#160;});,
<br>})<br>)| Property: Toolbar <br/><br/> @Html.EJS().Gantt("Gantt")<br>.Toolbar(new
List<string>() { "Add", "Cancel", "CollapseAll", "Delete", "Edit",
"ExpandAll", "NextTimeSpan",<br>&#160;&#160;"PrevTimeSpan", "Search", "Update" }).Render()
|
```

| Other toolbars | **Property:** *ToolbarSettings.toolbarItems*


```
@(Html.EJ().Gantt("Gantt")<br>.ToolbarSettings(toolbar =>
{<br>&#160;toolbar.ShowToolbar(true);<br>&#160;toolbar.ToolbarItems(new
List<GanttToolBarItems>()
{<br>&#160;&#160;GanttToolBarItems.Indent,<br>&#160;&#160;GanttToolBarItems.Outdent,<br>
&#160;&#160;GanttToolBarItems.CriticalPath,<br>&#160;&#160;GanttToolBarItems.ExcelExport,<br>&#160;&#160;GanttToolBarItems.PdfExport<br>});| Not applicable |
```

| Custom toolbar | **Property:** *ToolbarSettings.customToolBarItems*


```
@(Html.EJ().Gantt("Gantt")<br>.ToolbarSettings(toolbar =>
{<br>&#160;toolbar.ShowToolbar(true);<br>&#160;toolbar.CustomToolBarItems(ct =>
{<br>&#160;&#160;ct.TemplateID("#ColumnVisibility").TooltipText("Column
Visibility").Add();<br>&#160;&#160;ct.Text("Reset").TooltipText("Reset").Add();<br>&#160;});<br>})| Property: Toolbar <br/><br/> @<br>List<object> toolbarItems = new
List<object>();<br>toolbarItems.Add(new { text = "Quick Filter", tooltipText = "Quick Filter", id =
"toolbarfilter", align = "Right"
});<br>}<br><br>@Html.EJS().Gantt("Gantt")<br>.Toolbar(toolbarItems).Render() |
```

| Triggers when toolbar items clicked | **Event:** *ToolbarClick*

```
<br><br>@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
=><br>e.ToolbarClick("toolbarClick"))<br><br>Script:<br><script><br>function toolbarClick() {
}<br><script> | Event: ToolbarClick
<br><br>@Html.EJS().Gantt("Gantt")<br>.ToolbarClick("toolbarClick").Render()<br><br>Script:<br><script><br>function toolbarClick() { }<br><script> |
```

ToolTip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| To enable taskbar tooltip | **Property:** *EnableTaskbarTooltip*

 @(Html.EJ().Gantt("Gantt")
.EnableTaskbarTooltip(true)
)| **Property:**
TooltipSettings.showTooltip

 @Html.EJS().Gantt("Gantt")
.TooltipSettings(ts =>
 ts.ShowTooltip(true)).Render() |

| To define tooltip for all cells | **Property:** *CellTooltipTemplate*

 @(Html.EJ().Gantt("Gantt")
.CellTooltipTemplate("#CustomToolTip")
)| Not applicable |

| To define tooltip template for row drag action | **Property:** *DragTooltip*

 @(Html.EJ().Gantt("Gantt")
.DragTooltip(tooltip =>
 {
 tooltip.ShowTooltip(true);
})
)| Not applicable |

| To enable taskbar editing tooltip | **Property:** *EnableTaskbarDragTooltip*

 @(Html.EJ().Gantt("Gantt")
.EnableTaskbarDragTooltip(true)
)| Not Applicable |

| To enable/disable tooltip for grid cells | **Property:** *ShowGridCellTooltip*

 @(Html.EJ().Gantt("Gantt")
.ShowGridCellTooltip(true)
)| Not applicable |

| To enable/disable tooltip for grid cells | **Property:** *ShowGridExpandCellTooltip*

 @(Html.EJ().Gantt("Gantt")
.ShowGridExpandCellTooltip(true)
)| Not applicable |

| To define taskbar tooltip template in Gantt | **Property:** *TaskbarTooltipTemplate*

 @(Html.EJ().Gantt("Gantt")
TaskbarTooltipTemplate=" custom template tooltip
 string"
)| **Property:** *TooltipSettings.taskbar*

 @Html.EJS().Gantt("Gantt")
.TooltipSettings(ts => ts.Taskbar("custom template tooltip
 string")).Render() |

| To define taskbar tooltip template id in Gantt | **Property:** *TaskbarTooltipTemplateId*

 @(Html.EJ().Gantt("Gantt")
TaskbarTooltipTemplate="customtemplateID"
)| **Property:**
TooltipSettings.taskbar

 @Html.EJS().Gantt("Gantt")
.TooltipSettings(ts =>
 ts.Taskbar("#customtemplateID")).Render() |

| To define tooltip template for connector line | **Property:** *PredecessorTooltipTemplate*

 @(Html.EJ().Gantt("Gantt")
.PredecessorTooltipTemplate("ToolTipTemplate")
)|
Property: *TooltipSettings.connectorLine*

 @Html.EJS().Gantt("Gantt")
.TooltipSettings(ts =>
 ts.ConnectorLine("#dependencyLineTooltip")).Render() |

| To define template for progress resize tooltip | **Property:** *ProgressbarTooltipTemplate*

 @(Html.EJ().Gantt("Gantt")
.ProgressbarTooltipTemplate("progressbareeditingtooltiptemplat
 e")
)| **Property:** *TooltipSettings.editing*

 @Html.EJS().Gantt("Gantt")
.TooltipSettings(ts => ts.Editing("editingTooltip")).Render() |

| To define template id for progress resize tooltip | **Property:** *ProgressbarTooltipTemplateId*

 @(Html.EJ().Gantt("Gantt")
.ProgressbarTooltipTemplateId("progressbareeditingtooltiptempl
 D")
)| **Property:** *TooltipSettings.editing*

 @Html.EJS().Gantt("Gantt")
.TooltipSettings(ts => ts.Editing("#editingTooltip")).Render() |

| To define tooltip template for taskbar edit action | **Property:** *TaskbarEditingTooltipTemplate*


```
@(Html.EJ().Gantt("Gantt")<br>.TaskbarEditingTooltipTemplate("taskbareditingtooltiptemplate")<br>)| Property: TooltipSettings.editing <br><br>
@Html.EJS().Gantt("Gantt")<br>.TooltipSettings(ts => ts.Editing("editingTooltip")).Render() |
| To define tooltip template id for taskbar edit action | Property: TaskbarEditingTooltipTemplateId <br><br>
@(Html.EJ().Gantt("Gantt")<br>.TaskbarEditingTooltipTemplateId("taskbareditingtooltiptempID")<br>)| Property: TooltipSettings.editing <br><br>
@Html.EJS().Gantt("Gantt")<br>.TooltipSettings(ts => ts.Editing("#editingTooltip")).Render() |
```

Timeline

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

```
| To configure timeline settings in Gantt | Property: ScheduleHeaderSettings <br><br>
@(Html.EJ().Gantt("Gantt")<br>.ScheduleHeaderSettings(sh=>sh.WeekHeaderFormat("MMM dd , yyyy")<br>.DayHeaderFormat("ddd ")<br>.YearHeaderFormat("yyyy")<br>.MonthHeaderFormat("MMM yyyy")<br>.HourHeaderFormat("HH")<br>.ScheduleHeaderType(GanttScheduleHeaderType.Week)<br>.MinutesPerInterval(GanttMinutesPerInterval.FiveMinutes)<br>.WeekendBackground("#F2F2F2")<br>.TimescaleStartDateMode(GanttTimescaleRoundMode.Month)<br>.TimescaleUnitSize("70%")<br>.WeekStartDay(3)<br>.UpdateTimescaleView(false))<br>)|
Property: TimelineSettings
<br><br>@Html.EJS().Gantt("Gantt")<br>.TimelineSettings(ts=><br>ts.TimelineViewMode(Syncfusion.EJ2.Gantt<br>.TimelineViewMode.Week)<br>.&#160;.TimelineUnitSize(150)<br>.&#160;.WeekStartDay(3)<br>.&#160;.ShowTooltip(true)<br>.&#160;.WeekendBackground("#F2F2F2")<br>.&#160;.UpdateTimescaleView(false)<br>.&#160;.TopTier(tt =>
tt.Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Month)<br>.Format("MMM"))<br>.&#160;.BottomTier(bt => bt.Unit(Syncfusion.EJ2.Gantt.TimelineViewMode.Day))<br>.Render())|
| To define weekend background in Gantt | Property: WeekendBackground <br><br>
@(Html.EJ().Gantt("Gantt")<br>.WeekendBackground("rgba(116, 195, 231, 0.26)")<br>)| Not applicable |
| To highlight weekends | Property: HighlightWeekends <br><br>
@(Html.EJ().Gantt("Gantt")<br>.HighlightWeekends(true)<br>)| Property: HighlightWeekends
<br><br>@Html.EJS().Gantt("Gantt")<br>.HighlightWeekends(true).Render() |
| To include weekends | Property: IncludeWeekend <br><br>
@(Html.EJ().Gantt("Gantt")<br>.IncludeWeekend(true)<br>)| Property: IncludeWeekend
<br><br>@Html.EJS().Gantt("Gantt")<br>.IncludeWeekend(true).Render() |
| To define project start date in Gantt | Property: ScheduleStartDate
<br><br>@Html.EJ().Gantt("Gantt")<br>.ScheduleStartDate("02/01/2014")<br>)| Property: ProjectStartDate
<br><br>@Html.EJS().Gantt("Gantt")<br>.ProjectStartDate("03/24/2019").Render()|
```


| To define project end date in Gantt | **Property:** *ScheduleEndDate*

 @(Html.EJ().Gantt("Gantt")
.ScheduleEndDate("03/14/2014")
)| **Property:** *ProjectEndDate*

@Html.EJS().Gantt("Gantt")
.ProjectEndDate("04/28/2019").Render()|

| Update project start date and end date | **Method:** *updateScheduleDates(startDate, endDate)*

@(Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.updateScheduleDates("5/25/2017",
 "9/27/2017");
</script> | **Method:** *updateProjectDates(startDate, endDate, isTimelineRoundOff)*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.updateProjectDates("5/25/2017", "9/27/2017", true);
</script> |

Rows

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To enable/disable row drag and drop | **Property:** *AllowDragAndDrop*

 @(Html.EJ().Gantt("Gantt")
.AllowDragAndDrop(true)
)| Not applicable |

| To enable/disable alternate row background | **Property:** *EnableAltRow*

 @(Html.EJ().Gantt("Gantt")
.EnableAltRow(false)
)| Not applicable |

| To add Row height | **Property:** *RowHeight*

 @(Html.EJ().Gantt("Gantt")
.RowHeight(60)
)| **Property:** *RowHeight*

@Html.EJS().Gantt("Gantt")
.RowHeight(60).Render() |

| To render parent in collapsed state | **Property:** *EnableCollapseAll*

 @(Html.EJ().Gantt("Gantt")
.EnableCollapseAll(true)
)| **Property:** *CollapseAllParentTasks*

 @Html.EJS().Gantt("Gantt")
.CollapseAllParentTasks(true).Render() |

| Expand/collapse record by id | **Method:** *expandCollapseRecord(taskId)*

@(Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.expandCollapseRecord(1);
</script> | **Method:**
collapseById() *expandById()*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.expandById(1);

gant
 tObj.collapseById(1);
</script> |

| Expand all rows | **Method:** *expandAllItems()*

@(Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.expandAllItems();
</script> | **Method:** *expandAll()*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.expandAll();
</script> |

| Collapse all rows | **Method:** *collapseAllItems()*

@(Html.EJ().Gantt("Gantt")

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.collapseAllItems();
</script> | **Method:**
collapseAll()

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var
 ganttObj =
 document.getElementById('Gantt').ej2_instances[0];
ganttObj.collapseAll();
</script> |

| Triggers after row collapse action | **Event:** *Collapsed*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.Collapsed("collapsed"))

Script:
<script>
function collapsed() {
 }
<script> | **Event:** *Collapsed*

@(Html.EJS().Gantt("Gantt").Collapsed("collapsed").Render())

Script:
<scrip
 t>
function collapsed() { }
<script> |

| Triggers before row collapse action | **Event:** *Collapsing*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.Collapsing("collapsing"))

Script:
<script>
function collapsing() {
 }
<script> | **Event:** *Collapsing*

@(Html.EJS().Gantt("Gantt").Collapsing("collapsing").Render())

Script:
<scri
 pt>
function collapsing() { }
<script> |

| Triggers after Gantt row was expanded | **Event:** *Expanded*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.Expanded("expanded"))

Script:
<script>
function expanded() {
 }
<script> | **Event:** *Expanded*

@(Html.EJS().Gantt("Gantt").Expanded("expanded").Render())

Script:
<scrip
 t>
function expanded() { }
<script> |

| Triggers before Gantt row expand action | **Event:** *Expanding*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.Expanding("expanding"))

Script:
<script>
function expanding() {
 }
<script> | **Event:** *Expanding*

@(Html.EJS().Gantt("Gantt").Expanding("expanding").Render())

Script:
<scri
 pt>
function expanding() { }
<script> |

| Triggers before grid rows render action | **Event:** *RowDataBound*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.RowDataBound("rowDataBound"))

Script:
<script>
function
 rowDataBound() { }
<script> | **Event:** *RowDataBound*

@(Html.EJS().Gantt("Gantt").RowDataBound("rowDataBound").Render())

Script:
<script>
function rowDataBound() { }
<script> |

| Triggers while dragging a row | **Event:** *RowDrag*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.RowDrag("rowDrag"))

Script:
<script>
function rowDrag(event) { } | Not
 applicable |

| Triggers while while start to drag row | **Event:** *RowDragStart*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.RowDragStart("rowDragStart"))

Script:
<script>
function rowDrag(event)
 { } | Not applicable |

| Triggers while while drop a row | **Event:** *RowDragStop*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.RowDragStop("rowDragStop"))

Script:
<script>
function rowDrag(event)
 { } | Not applicable |

Resources

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To map resources | **Property:** *Resources*

@(Html.EJ().Gantt("Gantt")
.Resources(ViewBag.resource)
)| **Property:** *Resources*

@Html.EJS().Gantt("Gantt")
.Resources((IEnumerable<object>)ViewBag.projectResources)
.Render()|

| To map resource id field from resource collection | **Property:** *ResourceIdMapping*

@(Html.EJ().Gantt("Gantt")
.ResourceIdMapping("ResourceId")
)| **Property:**
ResourceIdMapping

@Html.EJS().Gantt("Gantt")
.ResourceIdMapping("ResourceId")
.Render() |

| To map resource name field from resource collection | **Property:** *ResourceNameMapping*

@(Html.EJ().Gantt("Gantt")
.ResourceNameMapping("ResourceName")
)| **Property:**
ResourceNameMapping

@Html.EJS().Gantt("Gantt")
.ResourceNameMapping("ResourceName")
.Render() |

| To map resource unit field from assigned resource collection | **Property:** *ResourceUnitMapping*

 @ (Html.EJ().Gantt("Gantt")
.ResourceUnitMapping("ResourceUnit")| Not applicable |

| To define resource view type of Gantt | **Property:** *ViewType*

@(Html.EJ().Gantt("Gantt")
.ViewType(GanttViewType.ResourceView)| Not applicable |

| To define mapping property for resource collection in resource view Gantt | **Property:**

ResourceCollectionMapping

@(Html.EJ().Gantt("Gantt")
.ResourceCollectionMapping("Resources")
)| Not Applicable |

| To map task collection from resources for resource view Gantt | **Property:** *TaskCollectionMapping*

 @ (Html.EJ().Gantt("Gantt")
.TaskCollectionMapping("Tasks")| Not applicable |

| To map group id for resource view Gantt | **Property:** *GroupIdMapping*

@(Html.EJ().Gantt("Gantt")
.GroupIdMapping("TeamId")
)| Not Applicable |

| To map group name for resource view Gantt | **Property:** *GroupNameMapping*

@(Html.EJ().Gantt("Gantt")
.GroupNameMapping("TeamName")
)| Not Applicable |

Baseline

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To render baseline | **Property:** *RenderBaseline*

@(Html.EJ().Gantt("Gantt")
.RenderBaseline(true)
)| **Property:** *RenderBaseline*

@Html.EJS().Gantt("Gantt")
.RenderBaseline(true).Render() |

| To define baselineColor | **Property:** *BaselineColor*

@(Html.EJ().Gantt("Gantt")
.BaselineColor("#fba41c")
)| **Property:** *BaselineColor*

@Html.EJS().Gantt("Gantt")
.BaselineColor("#fba41c")
.Render() |

Context Menu

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| To enable context menu | **Property:** *EnableContextMenu*

 @(Html.EJ().Gantt("Gantt")
.EnableContextMenu(true)
)| **Property:** *EnableContextMenu*

@Html.EJS().Gantt("Gantt")
.EnableContextMenu(true).Render() |

| To define custom menu items | **Event:** *ContextMenuOpen*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.ContextMenuOpen("contextMenuOpen"))

Script:
<script>
function
 contextMenuOpen()
 {
 event.contextMenuItems.push({
 headerText:
 "Expand/Collapse",
 menuId:
 "expand",
 iconPath: "url(Expand-02-
 WF.png)",
 eventHandler: function()
 {
 event handler for custom menu
 items
 }});
</script>
</script> | **Property:** *ContextMenuItems*

@
List<object> contextItems = new List<object>(); {
contextItems.Add(new {
 text = "Collapse the Row", target = ".e-content", id = "collapserow"
 });
contextItems.Add(new { text = "Expand the Row", target = ".e-content", id =
 "expandrow"
 });
}

@Html.EJS().Gantt("Gantt")
.ContextMenuItems(contextItems).Render() |

| Triggers before context menu opens | **Event:** *ContextMenuOpen*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.ContextMenuOpen("contextMenuOpen"))

Script:
<script>
function
 contextMenuOpen() { }
</script> | **Event:** *ContextMenuOpen*

@Html.EJS().Gantt("Gantt")
.ContextMenuOpen("contextMenuOpen").Render()

Script:
<script>
function contextMenuOpen() { }
</script> |

Scheduling Tasks

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| To define task scheduling mode in Gantt | **Property:** *TaskSchedulingMode*

 @(Html.EJ().Gantt("Gantt")
.TaskSchedulingMode(GanttTaskSchedulingMode.Auto)
)|
 Not applicable |

| To map task scheduling mode from data source | **Property:** *TaskSchedulingModeMapping*

 @(Html.EJ().Gantt("Gantt")
.TaskSchedulingModeMapping("TaskMode")
)| Not
 applicable |

| To enable schedule date validation while task predecessor draw action | **Property:**
ValidateManualTasksOnLinking

 @(Html.EJ().Gantt("Gantt")
.ValidateManualTasksOnLinking(false)
)| Not applicable |

| To define working time range of day | **Property:** *DayWorkingTime*

 @(Html.EJ().Gantt("Gantt")
.DayWorkingTime(dt => {
 dt.From("08:00
 AM").To("12:00 PM").Background("#EBF5FB").Add();
 })
)| **Property:**

DayWorkingTime

`@Html.EJS().Gantt("Gantt")
.DayWorkingTime(dw
=>{dw.From(9).To(18).Add();}).Render()` |

| To enable rounding off date value in taskbar editing | **Property:** *RoundOffDayworkingTime*

`@(Html.EJ().Gantt("Gantt")
.RoundOffDayworkingTime(true)
)|` Not applicable |

| To define non-working background color | **Property:** *NonWorkingBackground*

`@(Html.EJ().Gantt("Gantt")
.NonWorkingBackground("#B7C3D0")
)|` Not Applicable |

| To highlight non working time range in Gantt | **Property:** *HighlightNonWorkingTime*

`@(Html.EJ().Gantt("Gantt")
.HighlightNonWorkingTime(true)
)|` Not Applicable |

To set working days of a week | **Property:** *Workweek*

`@(Html.EJ().Gantt("Gantt")
.WorkWeek(new List<string>() {
"Sunday","Monday","Tuesday","Wednesday","Thursday" })
)|` **Property:** *WorkWeek*

`@Html.EJS().Gantt("Gantt")
.WorkWeek(new string[] { "Sunday", "Monday",
"Tuesday", "Wednesday", "Thursday" }).Render()` |

| To enable/disable Unscheduled tasks | **Property:** *AllowUnscheduledTask*

`@(Html.EJ().Gantt("Gantt")
.AllowUnscheduledTask(true)
)|` **Property:**
AllowUnscheduledTasks

`@Html.EJS().Gantt("Gantt")
.AllowUnscheduledTasks(true)
.Render()` |

Appearance and Customizations

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| To define taskbar background type in Gantt | **Property:** *TaskbarBackground*

`@(Html.EJ().Gantt("Gantt")
.TaskbarBackground("#1764d7")
)|` Not applicable |

| To define background color for parent taskbar | **Property:** *ParentTaskbarBackground*

`@(Html.EJ().Gantt("Gantt")
.ParentTaskbarBackground("#91dc88")
)|` Not applicable |

| To add Taskbar height | **Property:** *TaskbarHeight*

`@(Html.EJ().Gantt("Gantt")
.TaskbarHeight(30)
.RowHeight(40)
)|` **Property:**
TaskbarHeight

`@Html.EJS().Gantt("Gantt")
.RowHeight(60)
.TaskbarHeight(50).Render()` |

| To define background color for parent progress bar | **Property:** *ParentProgressbarBackground*

`@(Html.EJ().Gantt("Gantt")
.ParentProgressbarBackground("#af2f2f")
)|` Not
applicable |

| To define background color fro progress bar | **Property:** *ProgressbarBackground*

`@(Html.EJ().Gantt("Gantt")
.ProgressbarBackground("#8c83b1")
)|` Not applicable |

| To define height for progress bar | **Property:** *ProgressbarHeight*

`@(Html.EJ().Gantt("Gantt")
.ProgressbarHeight(80)
)|` Not Applicable |

| To render progress status taskbar | **Property:** *ShowProgressStatus*

`@(Html.EJ().Gantt("Gantt")
.ShowProgressStatus(true)
)|` **Property:**
LabelSettings.taskLabel

`@Html.EJS().Gantt("Gantt")
.LabelSettings(ls =>
ls.TaskLabel("${Progress}%")).Render()` |

| To set connectorline width | **Property:** *ConnectorLineWidth*

 @(Html.EJ().Gantt("Gantt")
.ConnectorLineWidth(3)
)| **Property:** *ConnectorLineWidth*

@Html.EJS().Gantt("Gantt")
.ConnectorLineWidth(3)
.Render()|

| To set connectorline background | **Property:** *ConnectorLineBackground*

 @(Html.EJ().Gantt("Gantt")
.ConnectorLineBackground("#0aecb8")
)| **Property:**
ConnectorLineBackground

@Html.EJS().Gantt("Gantt")
.ConnectorLineBackground("red").Render() |

| To define weekend background in Gantt | **Property:** *WeekendBackground*

 @(Html.EJ().Gantt("Gantt")
.WeekendBackground("rgba(116, 195, 231, 0.26)")
)| Not
 applicable |

| To define taskbar template | **Property:** *TaskbarTemplate*

 @(Html.EJ().Gantt("Gantt")
.TaskbarTemplate("#taskbarTemplate")
)| **Property:**
TaskbarTemplate

@Html.EJS().Gantt("Gantt")
.TaskbarTemplate("#TaskbarTemplate")
.Render() |

| To define parent taskbar template | **Property:** *ParentTaskbarTemplate*

 @(Html.EJ().Gantt("Gantt")
.ParentTaskbarTemplate("#ParentTaskbarTemplate")
)|
Property: *ParentTaskbarTemplate*

@Html.EJS().Gantt("Gantt")
.ParentTaskbarTemplate("#ParentTaskbarTemplate")

.Render()|

| To define milestone template | **Property:** *MilestoneTemplate*

 @(Html.EJ().Gantt("Gantt")
.MilestoneTemplate("#milestoneTemplate")
)| **Property:**
MilestoneTemplate

@Html.EJS().Gantt("Gantt")
.MilestoneTemplate("#milestoneTemplate")
.Render()
 er() |

| To define right task label | **Property:** *RightTaskLabelMapping*

 @(Html.EJ().Gantt("Gantt")
.RightTaskLabelMapping("TaskID")
)| **Property:**
LabelSettings.rightLabel

@Html.EJS().Gantt("Gantt")
.LabelSettings(ls =>
 ls.RightLabel("Task Name: \${taskData.TaskName}").Render()|

| To define left task label | **Property:** *LeftTaskLabelMapping*

 @(Html.EJ().Gantt("Gantt")
.LeftTaskLabelMapping("TaskName")
)| **Property:**
LabelSettings.leftLabel

@Html.EJS().Gantt("Gantt")
.LabelSettings(ls =>
 ls.LeftLabel("TaskId").Render()|

| To define right task label template | **Property:** *RightTaskLabelTemplate*

 @(Html.EJ().Gantt("Gantt")
.RightTaskLabelTemplate("#rightLabelTemplate")
)|
Property: *LabelSettings.rightLabel*

@Html.EJS().Gantt("Gantt")
.LabelSettings(ls =>
 ls.RightLabel("#RightTemplate").Render()|

| To define left task label template | **Property:** *LeftTaskLabelTemplate*

 @(Html.EJ().Gantt("Gantt")
.LeftTaskLabelTemplate("#leftLabelTemplate")
)| **Property:**
LabelSettings.leftLabel

@Html.EJS().Gantt("Gantt")
.LabelSettings(ls =>
 ls.LeftLabel("#LeftTemplate").Render()|

| To render resource names right to taskbar | **Property:** *ShowResourceNames*

 @(Html.EJ().Gantt("Gantt")
.ShowResourceNames(true)
)| **Property:**
LabelSettings.rightLabel

@Html.EJS().Gantt("Gantt")
.LabelSettings(Is =>
 Is.RightLabel(ResourceInfo).Render()) |

| To render task name left to taskbar | **Property:** *ShowTaskNames*

 @(Html.EJ().Gantt("Gantt")
[showTaskNames]="true"
)| **Property:** *LabelSettings.leftLabel*

@Html.EJS().Gantt("Gantt")
.LabelSettings("Is =>
 Is.LeftLabel("TaskName").Render()) |

| Triggers on taskbar rendering action | **Event:** *QueryTaskbarInfo*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.QueryTaskbarInfo("queryTaskbarInfo"))

Script:
<script>
function
 queryTaskbarInfo() {
<script> | **Event:** *QueryTaskbarInfo*

@Html.EJS().Gantt("Gantt")
.QueryTaskbarInfo("queryTaskbarInfo").Render()

Script:
<script>
function queryTaskbarInfo() { }
<script> |

| Triggers on grid cell rendering action | **Event:** *QueryCellInfo*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.QueryCellInfo("queryCellInfo"))

Script:
<script>
function queryCellInfo()
 { }
<script> | **Event:** *QueryCellInfo*

@Html.EJS().Gantt("Gantt")
.QueryCellInfo("queryCellInfo").Render()

Script:

<script>
function queryCellInfo() { }
<script> |

Stripline

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To define striplines | **Property:** *StripLines*

 @(Html.EJ().Gantt("Gantt")
.StripLines(new
 List<Syncfusion.JavaScript.Models.StripLine> {
new Syncfusion.JavaScript.Models.StripLine()
 {
 Day="03/02/2014",
 Label="Project
 Release",
 LineStyle="dotted",
 LineColor="blue",
 LineWidth=2,

 r}}
)| **Property:** *EventMarkers*

@Html.EJS().Gantt("Gantt")
.EventMarkers(em =>
 {
 em.Day("04/10/2019")
.Label("Project approval and kick-off")
.CssClass("e-
 custom-event-marker").Add();
}).Render() |

Holidays

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To define holidays | **Property:** *Holidays*

 @(Html.EJ().Gantt("Gantt")
.Holidays(new
 List<Syncfusion.JavaScript.Models.Holiday> {
new Syncfusion.JavaScript.Models.Holiday()
 {
 Day="2/03/2014",
 Background="YellowGreen",
 Label="Public
 holiday"
 }
}
)| **Property:** *Holidays*

@Html.EJS().Gantt("Gantt")
.Holidays(hol=>
 {
hol.From("04/11/2019")
.To("04/12/2019")
.Label("Public
 holidays")
.CssClass("e-custom-holiday").Add();
}).Render |

| To define days in holiday collection | **Property:** *Holidays.day*

 @(Html.EJ().Gantt("Gantt")
.Holidays(new List<Syncfusion.JavaScript.Models.Holiday>
 {
new Syncfusion.JavaScript.Models.Holiday()
 {
Day="2/03/2014"
},
})
)| **Property:** *Holidays.from*

@(Html.EJS().Gantt("Gantt")
..Holidays(hol=>
 {
hol.From("04/11/2019").Add();
}).Render()

Others

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To define height for Gantt | **Property:** *SizeSettings.height*

 @(Html.EJ().Gantt("Gantt")
.SizeSettings(size=>size.Height("350px"))
)| **Property:** *Height*

@(Html.EJS().Gantt("Gantt")
.Height("450px").Render())|

| To define width for Gantt | **Property:** *SizeSettings.width*

 @(Html.EJ().Gantt("Gantt")
.SizeSettings(size=>size.Width("700px"))
)| **Property:** *Width*

@(Html.EJS().Gantt("Gantt")
.Width("700px").Render())|

| To change splitter position | **Property:** *SplitterPosition*

 @(Html.EJ().Gantt("Gantt")
.SplitterPosition("50%")
)| Not applicable |

| To change splitter by position | **Property:** *SplitterSettings.Position*

 @(Html.EJ().Gantt("Gantt")
.SplitterSettings(sp=> sp.Position("60%"))
)| **Property:**
SplitterSettings.Position

 @Html.EJS().Gantt("Gantt")
.SplitterSettings(ss=>ss.Position("80%")).Render())|

| To change splitter by index | **Property:** *SplitterSettings.Index*

 @(Html.EJ().Gantt("Gantt")
.SplitterSettings(sp=> sp.Index(3))
)| **Property:**
SplitterSettings.ColumnIndex

 @Html.EJS().Gantt("Gantt")
.SplitterSettings(ss=>ss.ColumnIndex(3)).Render())|

| To define resource view type of Gantt | **Property:** *ViewType*

 @(Html.EJ().Gantt("Gantt")
.ViewType(GanttViewType.ProjectView)| Not applicable |

| To enable Localization | **Property:** *Locale*

 @(Html.EJ().Gantt("Gantt")
.Locale("fr-FR")
)| **Property:** *Locale*

@Html.EJS().Gantt("Gantt")
.Locale("fr-FR").Render()) |

| To specify the date format for Gantt | **Property:** *DateFormat*

 @(Html.EJ().Gantt("Gantt")
.DateFormat("M/d/yyyy hh:mm:ss tt")
)| **Property:**
DateFormat

@Html.EJS().Gantt("Gantt")
.DateFormat("M/d/yyyy hh:mm:ss
 tt").Render()) |

| To enable/disable key navigation | **Property:** *AllowKeyboardNavigation*

 @(Html.EJ().Gantt("Gantt")
.AllowKeyboardNavigation(true)
)| **Property:** *AllowKeyboard*

 @Html.EJS().Gantt("Gantt")
.AllowKeyboard(true).Render())|

| To enable serial number support | **Property:** *EnableSerialNumber*

 @(Html.EJ().Gantt("Gantt")
.EnableSerialNumber(true)
)| Not applicable |

| To enable/disable predecessor validation | **Property:** *EnablePredecessorValidation*

 @(Html.EJ().Gantt("Gantt")
.EnablePredecessorValidation(false)
)| **Property:**

EnablePredecessorValidation

@Html.EJS().Gantt("Gantt")
.EnablePredecessorValidation(false).Render() |

| To set timescale for working hours | **Property:** *WorkingTimeScale*

@(Html.EJ().Gantt("Gantt")
.WorkingTimeScale(GanttWorkingTimeScale.TimeScale24Hours)

) | **Property:** *DayWorkingTime*

@Html.EJS().Gantt("Gantt")
.DayWorkingTime(dw =>{
dw.From(9).To(18).Add();
}).Render() |

| To enable work break down structure in Gantt | **Property:** *EnableWBS*

@(Html.EJ().Gantt("Gantt")
.EnableWBS(true)
) | Not Applicable |

| To enable work break down structure predecessor in Gantt | **Property:** *EnableWBSPredecessor*

 @ (Html.EJ().Gantt("Gantt")
.EnableWBSPredecessor(true)
) | Not Applicable |

| To map work value from data source | **Property:** *WorkMapping*

@(Html.EJ().Gantt("Gantt")
.WorkMapping("estimatedWork")
) | Not applicable |

| To define work unit for tasks | **Property:** *WorkUnit*

@(Html.EJ().Gantt("Gantt")
.WorkUnit(GanttWorkUnit.Minute)
) | Not applicable |

| To define task type in Gantt | **Property:** *TaskType*

@(Html.EJ().Gantt("Gantt")
.TaskType(GanttTaskType.FixedWork)
) | Not applicable |

| To enable/disable multiple exporting option | **Property:** *AllowMultipleExporting*

@(Html.EJ().Gantt("Gantt")
.AllowMultipleExporting(true)
) | Not applicable |

| To enable virtual rendering in Gantt | **Property:** *EnableVirtualization*

@(Html.EJ().Gantt("Gantt")
.EnableVirtualization(true)
) | Not Applicable |

| Change splitter position dynamically | **Method:** *SetSplitterIndex(Index) SetSplitterPosition(Width)*

 @ (Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
\$(("#Gantt").data("ejGantt"));
ganttObj.setSplitterIndex(3);

ganttObj.setSplitterPositi
on("40%");
</script> | **Method:** *setSplitterPosition(value,type)*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
ganttObj.setSplitterPosition('40%',
'position');

ganttObj.setSplitterPosition(3, 'columnIndex'); |

| To destroy Gantt | **Method:** *destroy()*

@(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
\$(("#Gantt").data("ejGantt"));
ganttObj.destroy();
</script> | **Method:** *destroy()*

@Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
ganttObj.destroy();
</script> |

| To update task id | **Method:** *updateTaskId(currentId, newId)*

@(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
\$(("#Gantt").data("ejGantt"));
ganttObj.updateTaskId(5, 7);
</script> | Not applicable |

| To set scroll top for Gantt | **Method:** *SetScrollTop(top)*

@(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
\$(("#Gantt").data("ejGantt"));
ganttObj.setScrollTop(50);
</script> | **Method:**
SetScrollTop()

 @Html.EJS().Gantt("Gantt").Render()

Script:
<script>
var

```
ganttObj =
document.getElementById('Gantt').ej2_instances[0];<br>ganttObj.setScrollTop(200);<br></script> |
```

| To get columns to edit in resource view | **Method:** *getResourceViewEditColumns()*

 @(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
columns =
 ganttObj.getResourceViewEditColumns();
</script> | Not applicable |

| Show/hide critical path in Gantt | **Method:** *showCriticalPath(isShown)*

@(Html.EJ().Gantt("Gantt"))

Script:
<script>
var ganttObj =
 \$("#Gantt").data("ejGantt");
ganttObj.showCriticalPath(true);
</script> | Not applicable |

| Triggers on initialization of Gantt control | **Event:** *Load*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.Load("load"))

Script:
<script>
function load() { }
</script> | **Event:** *Load*

@Html.EJS().Gantt("Gantt").Load("load").Render()

Script:
<script>
func
 tion load() { }
</script> |

| Triggers after splitter resize action | **Event:** *SplitterResized*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.SplitterResized("splitterResized"))

Script:
<script>
function
 splitterResized() { }
</script> | **Event:** *SplitterResized*

@Html.EJS().Gantt("Gantt").SplitterResized("splitterResized").Render()

**Sc
 ript:**
<script>
function splitterResized() { }
</script> |

| Triggers when taskbar item is clicked | **Event:** *TaskbarClick*

@(Html.EJ().Gantt("Gantt").ClientSideEvents(e
 =>
e.TaskbarClick("taskbarClick"))

Script:
<script>
function
 taskbarClick(event) { } | Not applicable |

How To

Open add/edit dialog dynamically

Gantt add and edit dialogs can be opened dynamically by using `openAddDialog` and `openEditDialog` methods. The following code example shows how to open add and dialog on separate button click actions.

CSHTML

```
@Html.EJS().Button("editDialog").Content("Open Edit  
Dialog").CssClass("e-primary").Render()
@Html.EJS().Button("addDialog").Content("Open Add Dialog").CssClass("e-  
primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc  
e).Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P  
rogress("Progress").Child("SubTasks").EditSettings(es =>
es.AllowEditing(true).AllowAdding(true)).Render()

<script>
```

```

        document.getElementById('editDialog').addEventListener('click',
function (args) {
    var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
    ganttObj.editModule.dialogModule.openEditDialog();
});
        document.getElementById('addDialog').addEventListener('click',
function (args) {
    var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
    ganttObj.editModule.dialogModule.openAddDialog();
});
</script>

```

OPENEDITADDDIALOG.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}

```

Note: We should select any one of the row in Gantt to open the edit dialog.

Set the vertical scroll position

In the Gantt control, you can set the vertical scroller position dynamically by clicking the custom button using the `setScrollTop` method.

CSHTML

```

@Html.EJS().Button("setScrollTop").Content("Change Scroll
Position").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").Child("SubTasks")
).Render()

<script>
    document.getElementById('setScrollTop').addEventListener('click',
function (args) {
    var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
    ganttObj.ganttChartModule.scrollObject.setScrollTop(300);
});
</script>

```

SETSCROLLTOP.CS

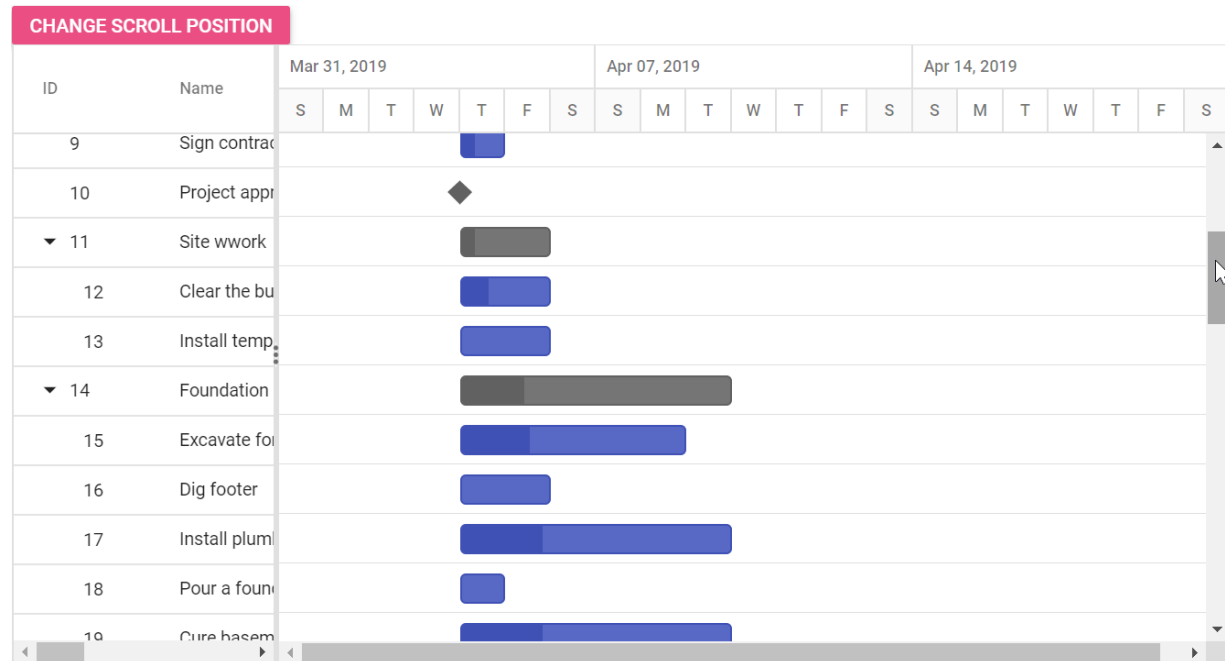
```

public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}

```

```
}

```



Change schedule start and end dates

In the Gantt control, you can change the schedule start and end dates by clicking the custom button programmatically using the `updateProjectDates` method. You can pass the start and end dates as method arguments to the `updateProjectDates` method. You can also pass the Boolean value as an additional parameter, which is used to round-off the schedule start and end dates displayed in Gantt timeline.

CSHTML

```
@Html.EJS().Button("updateSchedule").Content("Update
Schedule").CssClass("e-primary").Render()

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").Child("SubTasks")
).Render()

<script>

document.getElementById("updateSchedule").addEventListener('click', () => {
    var ganttObj =
document.getElementById("Gantt").ej2_instances[0];
    ganttObj.updateProjectDates(new Date('01/10/2019'), new
Date('06/20/2019'), true);
});

</script>
```

CHANGESCHEDULEDATES.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}
```

Copy and Paste records in Gantt

You can copy and paste a record in the Gantt chart by using the `addRecord` method and `custom context menu`. It is also possible to copy and paste the parent record with multiple hierarchical child records on the required position.

CSHTML

```
@{
    List<object> contextItems = new List<object> { "AutoFitAll", "AutoFit",
    "TaskInformation", "DeleteTask", "Save", "Cancel",
        "SortAscending", "SortDescending", "Add", "DeleteDependency",
    "Convert"};
    contextItems.Add(new { text= "Copy", target= ".e-content", id= "copy"
    });
    contextItems.Add(new { text= "Paste", target= ".e-content", id= "paste"
    });
}

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName"

).StartDate("StartDate").Duration("Duration").Progress("Progress").Dependenc
y("Predecessor").Child("SubTasks")
).EnableContextMenu(true).EditSettings(es =>
es.AllowAdding(true).AllowEditing(true).AllowDeleting(true)

).ContextMenuItems(contextItems).ContextMenuOpen("ContextMenuOpen").ContextM
enuClick("ContextMenuClick").Render()
<script>
    var copiedRecord;
    function ContextMenuOpen(args) {
        if (args.type !== 'Header') {
            if (copiedRecord) {
                args.hideItems.push('Copy');
            } else {
                args.hideItems.push('Paste');
            }
        }
    }
    function ContextMenuClick(args) {
        var gantt = document.getElementById("Gantt").ej2_instances[0];
        if (args.item.id === 'copy') {
            copiedRecord = args.rowData;
            copiedRecord.taskData.TaskID = gantt.currentViewData.length + 1;
        }
        if (args.item.id === 'paste') {
            gantt.addRecord(copiedRecord.taskData, 'Below', args.rowData.index);
            if (copiedRecord.hasChildRecords) {
                addChildRecords(copiedRecord, args.rowData.index + 1);
            }
        }
    }
}
```

```

    }
    copiedRecord = undefined;
  }
}
function addChildRecords(record, index) {
  for(var i=0; i<record.childRecords.length; i++) {
    var childRecord = record.childRecords[i];
    childRecord.taskData.TaskID = gantt.currentViewData.length + 1;
    gantt.addRecord(childRecord.taskData, 'Child', index);
    if(childRecord.hasChildRecords) {
      addChildRecords(childRecord, index + (i+1));
    }
  }
}
}
</script>

```

COPYPASTERECORDS.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = ProjectNewData();
    return View();
}

```

Updating row drag and dropped index position on server side

Row dropped record's index position can be maintained in the Gantt chart by changing the database table index position using the `rowDrop` event. In this event, the `fromIndex` and `dropIndex` values can be passed to the server side using Ajax request. On the server side, the `insert` and `insertAtTop` methods are used to update the row index position. The following code snippets explain the solution.

CSHTML

```

@Html.EJS().Gantt("DragAndDrop").DataSource(dataManager =>
{
    dataManager.Url("https://localhost:44379/Home/UrlDatasource").Adaptor("UrlAdaptor")
}).AllowRowDragAndDrop(true).Height("450px").TreeColumnIndex(1).TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Dependency("Predecessor").Child("SubTasks")).Render()
<script>
    function rowDrop(args) {
        var record = this.flatData[args.fromIndex][this.taskFields.id];
        var record2 = this.flatData[args.dropIndex][this.taskFields.id];
        var data = args.data[0];
        var uri = 'https://localhost:44379/Home/RowDropMethod';
        var dragdropdata = {
            record: data[0].taskData,
            position: args.dropIndex,
            dragidMapping: record,
            dropidMapping: record2
        };
        var ajax = new Ajax(

```

```

        {
            url: uri,
            type: 'POST',
            contentType: "application/json",
            data: JSON.stringify(dragdropdata),
        });
        ajax.send();
    }
}
</script>

```

MAINTAINRECORDINDEX.CS

```

public IActionResult RowDropMethod([FromBody] DragDropData value)
{
    var data = new CRUDModel();
    copyRecord = true;
    if (value.position == "bottomSegment" || value.position ==
"topSegment")
    {
        {
            var childCount = 0;
            int parent1 = (int)value.record.parentID;
            childCount += FindChildRecords(parent1);
            if (childCount == 1)
            {
                var i = 0;
                for (; i < DataList.Count; i++)
                {
                    if (DataList[i].taskID == parent1)
                    {
                        DataList[i].isParent = false;
                        break;
                    }
                    if (DataList[i].taskID == value.record.taskID)
                    {
                        DataList[i].parentID = null;
                        break;
                    }
                }
            }
            DataList.Remove(DataList.Where(ds => ds.taskID ==
value.dragidMapping).FirstOrDefault());
            var j = 0;
            for (; j < DataList.Count; j++)
            {
                if (DataList[j].taskID == value.dropidMapping)
                {
                    value.record.parentID = DataList[j].parentID;
                    break;
                }
            }
            data.Value = value.record;
            if (value.position == "bottomSegment")
            {

```

```

        this.Insert(data, value.dropidMapping);
    }
    else if (value.position == "topSegment")
    {
        this.InsertAtTop(data, value.dropidMapping);
    }
}
else if (value.position == "middleSegment")
{
    DataList.Remove(DataList.Where(ds => ds.taskID ==
value.dragidMapping).FirstOrDefault());
    value.record.parentID = value.dropidMapping;
    FindDropdata(value.dropidMapping);
    data.Value = value.record;
    this.Insert(data, value.dropidMapping);
}
copyRecord = false;
return Json(new { updatedRecords = value.record });
}

```

Add Custom Fields in the General Tab in Add/Edit Dialog

Generally in Gantt, Custom fields are displayed in the Custom Tab of the Add/Edit dialogs. However, they can be included in the General Tab of Add/Edit Dialog Box using `actionBegin` and `actionComplete` events. These events are used to append the custom field to the dialog box. The following code snippets demonstrate the solution.

CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name(
"TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").P
rogress("Progress").Child("SubTasks")).EditDialogFields(edf =>
{
    edf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.General).HeaderText("General")
.Add();
    edf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Dependency).Add();
    edf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Resources).Add();
    edf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Notes).Add();
}).AddDialogFields(adf=> {
    adf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.General).HeaderText("General
Tab").Add();
    adf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Dependency).Add();
    adf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Resources).Add();
    adf.Type(Syncfusion.EJ2.Gantt.DialogFieldType.Notes).Add();
}).Columns(col =>
{
    col.Field("TaskId").Width(150).Add();
    col.Field("TaskName").HeaderText("Job
Name").Width(250).Add();
    col.Field("StartDate").Width(250).Add();
    col.Field("EndDate").Width(250).Add();
    col.Field("Duration").Width(250).Add();
}

```



```

        col.Field("Progress").Width(250).Add();
        col.Field("CustomField").Width(250).Add();
    }).Toolbar(new List<string>()
    { "Add", "Cancel", "CollapseAll", "Delete", "Edit", "ExpandAll",
    "NextTimeSpan", "PrevTimeSpan", "Search", "Update" }).EditSettings(es=>
es.AllowAdding(true).AllowEditing(true).Mode(Syncfusion.EJ2.Gantt.EditMode.D
ialog)).ActionBegin(
    "actionBegin").ActionComplete("actionComplete").Render()
<script>
    var divElement;
    var inputs = {
        booleanedit: ej.buttons.CheckBox,
        dropdownedit: ej.dropdowns.DropDownList,
        datepickeredit: ej.calendars.DatePicker,
        datetimepickeredit: ej.calendars.DateTimePicker,
        maskededit: ej.inputs.MaskedTextBox,
        numericedit: ej.inputs.NumericTextBox,
        stringedit: ej.inputs.TextBox
    };
    function actionBegin(args) {
        var ganttObj = document.getElementById("Gantt").ej2_instances[0];
        if (args.requestType === "beforeOpenEditDialog" || args.requestType
=== "beforeOpenAddDialog") {
            var column = ganttObj.columnByField["CustomField"];
            divElement = ganttObj.createElement("div", {
                className: "e-edit-form-column"
            });
            var inputElement;
            inputElement = ganttObj.createElement("input", {
                attrs: {
                    type: "text",
                    id: this.controlId + "" + column.field,
                    name: column.field,
                    title: column.field
                }
            });
            divElement.appendChild(inputElement);
            var input = {
                enabled: true,
                floatLabelType: "Auto",
                placeholder: "CustomField",
                value: args.rowData.CustomField
            };
            var inputObj = new inputs[column.editType](input);
            inputObj.appendTo(inputElement);
        }
        function actionComplete(args) {
            var ganttObj = document.getElementById("Gantt").ej2_instances[0];
            if (args.requestType === "openEditDialog" || args.requestType ===
"openAddDialog") {
                var generalTab = document.getElementById(
                    ganttObj.controlId + "GeneralTabContainer"
                );
                generalTab.appendChild(divElement);
            }
        }
    }

```

```
}
</script>
```

CUSTOMFIELD.CS

```
public ActionResult Index()
{
    ViewBag.DataSource = ProjectNewData();
    return View();
}
```

Maintain zoomToFit

In the Gantt control, While performing edit actions or dynamically change dataSource, the timeline gets refreshed. When zoomToFit toolbar item is clicked and perform editing actions or dynamically change dataSource, the timeline gets refreshed. So that, the timeline will not fit to the project any more.

Maintain zoomToFit after edit actions

We can maintain zoomToFit after editing actions(cell edit,dialog edit,taskbar edit) by using fitToProject method in actionComplete and taskbarEdited event.

CSHTML

```
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.ActionComplete("actionComplete").TaskbarEdited("taskbarEdited").Height("450px")
.TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration")
.Progress("Progress").Dependency("Predecessor").Child("SubTasks").Toolbar(new List<string>()
{"Edit", "ZoomToFit"}).LabelSettings(ls =>
ls.LeftLabel("TaskName").EditSettings(es =>

es.AllowTaskbarEditing(true).AllowEditing(true)).ProjectStartDate("03/24/2019").ProjectEndDate(
"04/28/2019").Render()

<script>
    function actionComplete(args) {
        if ((args.action === "CellEditing" || args.action === "DialogEditing") && args.requestType === "save") {
            var obj = document.getElementById("Gantt").ej2_instances[0];
            obj.dataSource = '@(ViewBag.DataSource)';
            obj.fitToProject();
        }
    }
    function taskbarEdited(args) {
        if (args) {
            var obj = document.getElementById("Gantt").ej2_instances[0];
            obj.dataSource = '@(ViewBag.DataSource)';
            obj.fitToProject();
        }
    }
</script>
```

MAINTAINZOOMTOFIT.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = ganttData();
    return View();
}

public static List<GanttDataSource> ganttData()
{
    List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
    GanttDataSource Record1 = new GanttDataSource()
    {
        TaskId = 1,
        TaskName = "Project initiation",
        StartDate = new DateTime(2019, 04, 02),
        EndDate = new DateTime(2019, 04, 21),
        SubTasks = new List<GanttDataSource>(),
    };
    GanttDataSource Child1 = new GanttDataSource()
    {
        TaskId = 2,
        TaskName = "Identify site location",
        StartDate = new DateTime(2019, 04, 02),
        Duration = 4,
        Progress = 70,
    };
    GanttDataSource Child2 = new GanttDataSource()
    {
        TaskId = 3,
        TaskName = "Perform soil test",
        StartDate = new DateTime(2019, 04, 02),
        Duration = 4,
        Progress = 50,
        Predecessor = "2FS"
    };
    GanttDataSource Child3 = new GanttDataSource()
    {
        TaskId = 4,
        TaskName = "Soil test approval",
        StartDate = new DateTime(2019, 04, 02),
        Duration = 4,
        Progress = 50,
        Predecessor = "3FS"
    };
    Record1.SubTasks.Add(Child1);
    Record1.SubTasks.Add(Child2);
    Record1.SubTasks.Add(Child3);
    GanttDataSource Record2 = new GanttDataSource()
    {
        TaskId = 5,
        TaskName = "Project estimation",
        StartDate = new DateTime(2019, 04, 02),
        EndDate = new DateTime(2019, 04, 21),
        SubTasks = new List<GanttDataSource>()
    };
    GanttDataSource Child4 = new GanttDataSource()
    {
        TaskId = 6,
```

```

        TaskName = "Develop floor plan for estimation",
        StartDate = new DateTime(2019, 04, 04),
        Duration = 3,
        Progress = 70
    };
    GanttDataSource Child5 = new GanttDataSource()
    {
        TaskId = 7,
        TaskName = "List materials",
        StartDate = new DateTime(2019, 04, 04),
        Duration = 3,
        Progress = 50,
        Predecessor = "6SS"
    };
    Record2.SubTasks.Add(Child4);
    Record2.SubTasks.Add(Child5);
    GanttDataSourceCollection.Add(Record1);
    GanttDataSourceCollection.Add(Record2);
    return GanttDataSourceCollection;
}

public class GanttDataSource
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public int? Duration { get; set; }
    public int Progress { get; set; }
    public string Predecessor { get; set; }
    public List<GanttDataSource> SubTasks { get; set; }
}

```

Maintain zoomToFit after change dataSource dynamically

We can maintain `zoomToFit` after change dataSource dynamically, by calling `fitToProject` method in dataBound event.

CSHTML

```

@Html.EJS().Button("changedata").Content("Change Data").CssClass("e-
primary").Render()
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSourc
e).DataBound("dataBound").Height("450px").TaskFields(ts =>

ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").D
uration("Duration").Progress("Progress").Dependency(
    "Predecessor").Child("SubTasks").Toolbar(new List<string>()
    {"ZoomToFit"}).LabelSettings(ls =>
ls.LeftLabel("TaskName").ProjectStartDate("03/24/2019").ProjectEndDate(
    "04/28/2019").Render()

<script>
    document.getElementById('changedata').addEventListener('click',
function (args) {
    var ganttObj =
document.getElementById('Gantt').ej2_instances[0];
    ganttObj.dataSource = '@(ViewBag.Data)';
});

```

```
function dataBound(args) {  
    var ganttObj =  
document.getElementById('Gantt').ej2_instances[0];  
    obj.fitToProject();  
}  
</script>
```

MAINTAINZOOMTOFITDATASOURCE.CS

```
public IActionResult Index()  
{  
    ViewBag.DataSource = ganttData();  
    ViewBag.Data = ProjectNewData();  
    return View();  
}  
public static List<GanttDataSource> ganttData()  
{  
    List<GanttDataSource> GanttDataSourceCollection = new  
List<GanttDataSource>();  
    GanttDataSource Record1 = new GanttDataSource()  
    {  
        TaskId = 1,  
        TaskName = "Project initiation",  
        StartDate = new DateTime(2019, 04, 02),  
        EndDate = new DateTime(2019, 04, 21),  
        SubTasks = new List<GanttDataSource>(),  
    };  
    GanttDataSource Child1 = new GanttDataSource()  
    {  
        TaskId = 2,  
        TaskName = "Identify site location",  
        StartDate = new DateTime(2019, 04, 02),  
        Duration = 4,  
        Progress = 70,  
    };  
    GanttDataSource Child2 = new GanttDataSource()  
    {  
        TaskId = 3,  
        TaskName = "Perform soil test",  
        StartDate = new DateTime(2019, 04, 02),  
        Duration = 4,  
        Progress = 50,  
        Predecessor = "2FS"  
    };  
    GanttDataSource Child3 = new GanttDataSource()  
    {  
        TaskId = 4,  
        TaskName = "Soil test approval",  
        StartDate = new DateTime(2019, 04, 02),  
        Duration = 4,  
        Progress = 50,  
        Predecessor = "3FS"  
    };  
    Record1.SubTasks.Add(Child1);  
    Record1.SubTasks.Add(Child2);  
    Record1.SubTasks.Add(Child3);  
}
```

```

GanttDataSource Record2 = new GanttDataSource()
{
    TaskId = 5,
    TaskName = "Project estimation",
    StartDate = new DateTime(2019, 04, 02),
    EndDate = new DateTime(2019, 04, 21),
    SubTasks = new List<GanttDataSource>()
};
GanttDataSource Child4 = new GanttDataSource()
{
    TaskId = 6,
    TaskName = "Develop floor plan for estimation",
    StartDate = new DateTime(2019, 04, 04),
    Duration = 3,
    Progress = 70
};
GanttDataSource Child5 = new GanttDataSource()
{
    TaskId = 7,
    TaskName = "List materials",
    StartDate = new DateTime(2019, 04, 04),
    Duration = 3,
    Progress = 50,
    Predecessor = "6SS"
};
Record2.SubTasks.Add(Child4);
Record2.SubTasks.Add(Child5);
GanttDataSourceCollection.Add(Record1);
GanttDataSourceCollection.Add(Record2);
return GanttDataSourceCollection;
}
public class GanttDataSource
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public int? Duration { get; set; }
    public int Progress { get; set; }
    public string Predecessor { get; set; }
    public List<GanttDataSource> SubTasks { get; set; }
}
public static List<GanttDataSource> ProjectNewData()
{
    List<GanttDataSource> GanttDataSourceCollection = new
List<GanttDataSource>();
    GanttDataSource Record1 = new GanttDataSource()
    {
        TaskId = 1,
        TaskName = "Product concept",
        StartDate = new DateTime(2019, 04, 02),
        EndDate = new DateTime(2019, 04, 21),
        SubTasks = new List<GanttDataSource>(),
    };
    GanttDataSource Child1 = new GanttDataSource()
    {
        TaskId = 2,

```

```
        TaskName = "Defining the product and its usage",
        StartDate = new DateTime(2019, 04, 02),
        Progress = 30,
        Duration = 3,
    };
    GanttDataSource Child2 = new GanttDataSource()
    {
        TaskId = 3,
        TaskName = "Defining target audience",
        StartDate = new DateTime(2019, 04, 02),
        Duration = 3,
        Predecessor = "2FS"
    };
    GanttDataSource Child3 = new GanttDataSource()
    {
        TaskId = 4,
        TaskName = "Prepare product sketch and notes",
        StartDate = new DateTime(2019, 04, 02),
        Progress = 30,
        Duration = 2,
        Predecessor = "3FF"
    };
    Record1.SubTasks.Add(Child1);
    Record1.SubTasks.Add(Child2);
    Record1.SubTasks.Add(Child3);
    GanttDataSource Record2 = new GanttDataSource()
    {
        TaskId = 5,
        TaskName = "Concept approval",
        StartDate = new DateTime(2019, 04, 02),
        Duration = 0,
        SubTasks = new List<GanttDataSource>(),
    };
    GanttDataSource Child4 = new GanttDataSource()
    {
        TaskId = 6,
        TaskName = "Demand analysis",
        StartDate = new DateTime(2019, 04, 04),
        EndDate = new DateTime(2019, 04, 21),
        SubTasks = new List<GanttDataSource>(),
    };
    GanttDataSource Child5 = new GanttDataSource()
    {
        TaskId = 7,
        TaskName = "Customer strength",
        StartDate = new DateTime(2019, 04, 04),
        Duration = 4,
        Progress = 30,
        Predecessor = "6SS"
    };
    Record2.SubTasks.Add(Child4);
    Record2.SubTasks.Add(Child5);
    GanttDataSourceCollection.Add(Record1);
    GanttDataSourceCollection.Add(Record2);
    return GanttDataSourceCollection;
}
```

Drag and Drop the Record from another component to Gantt

In Gantt, it is possible to drag a record from another component and drop it in Gantt chart with updating the Gantt record. Here, dragging an item from **TreeView** component to Gantt and that item is updated as a resource for the Gantt record, we can achieve this, by using [nodeDragStop](#) event of **TreeView** control.

CSHTML

```
@(Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("280px").AllowSelection(true)
.TaskFields(ts =>
ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")

.Dependency("Predecessor").Child("SubTasks").ResourceInfo("Resources")
.EditSettings(es => es.AllowEditing(true))
.Resources((IEnumerable<object>)ViewBag.Resources)
.ResourceFields(rf => rf.Id("ResourceId").Name("ResourceName"))
.LabelSettings(ls => ls.RightLabel("Resources"))
.Render()
)
)
@ (Html.EJS().TreeView("treedata").AllowDragAndDrop(true).Fields(field =>
field.Id("ResourceId").Text("ResourceName").DataSource(ViewBag.Resources)
.NodeDragStop("nodeDragStop")
.Render()
)
)
<script>
function nodeDragStop(args) {
    var ganttChart = document.getElementById('Gantt').ej2_instances[0];
    var chartEle = ej.base.closest(args.target, '.e-chart-row');
    var gridEle = ej.base.closest(args.target, '.e-row');
    if (gridEle) {
        var index = ganttChart.treeGrid.getRows().indexOf(gridEle);
        ganttChart.selectRow(index);
    }
    if (chartEle) {
        var index = chartEle.ariaRowIndex;
        ganttChart.selectRow(Number(index));
    }
    var record = args.draggedNodeData;
    var selectedData = ganttChart.flatData[ganttChart.selectedRowIndex];
    var selectedDataResource = selectedData.taskData.Resources;
    var resources = [];
    if (selectedDataResource) {
        for (var i = 0; i < selectedDataResource.length; i++) {
            resources.push(selectedDataResource[i].ResourceId);
        }
    }
    resources.push(parseInt(record.id));
    if (chartEle || gridEle) {
        var data = {
            TaskId: selectedData.taskData.TaskId,
            Resources: resources
        }
    }
}
```



```

    };
    ganttChart.updateRecordByID(data);
    var elements = document.querySelectorAll('.e-drag-item');
    while (elements.length > 0 && elements[0].parentNode) {
        elements[0].parentNode.removeChild(elements[0]);
    }
}
}
</script>

```

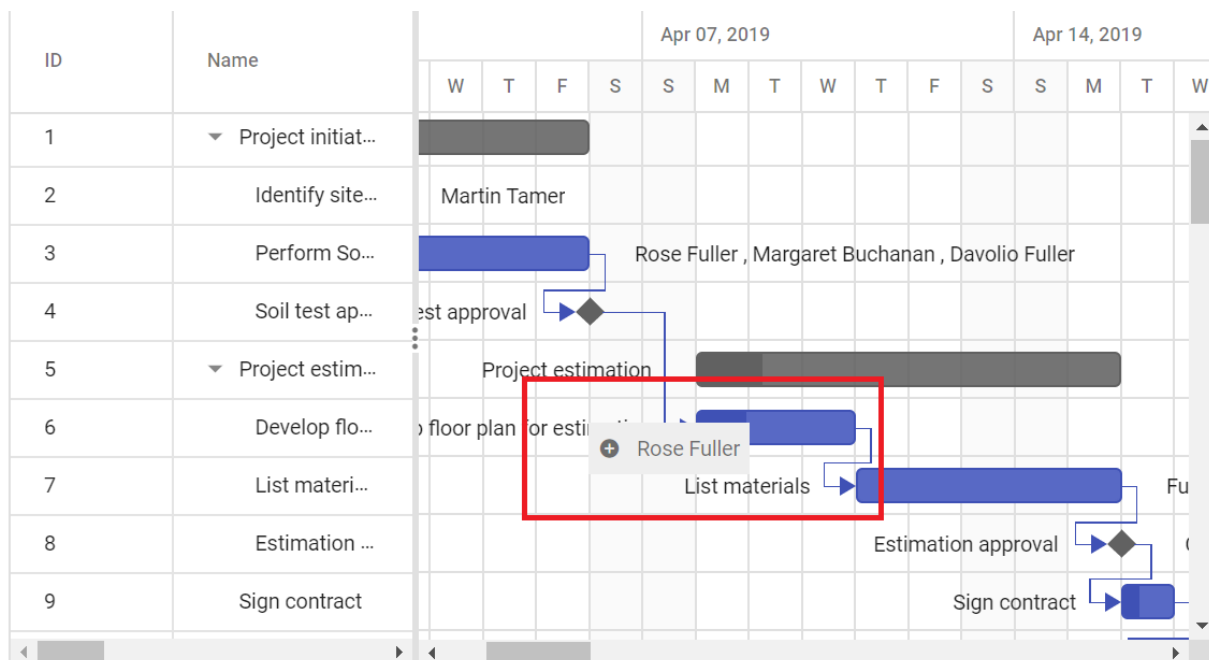
DRAGANDDROP.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    ViewBag.projectResources = projectResources();
    return View();
}

```

The following screenshot shows dropping record from another component in to Gantt, and **Rose Fuller** is added as resource for the task **Develop floor plan estimation**.



Flat list

Martin Tamer
Rose Fuller
Margaret Buchanan
Fuller King

Set new row position in Gantt

In Gantt, a new row can be added in one of the following positions: Top, Bottom, Above, Below and Child. This position can be specified through the `newRowPosition` property. We can make use of the `toolbarClick` event to create a context menu that specifies the position in which the new row is to be added when adding a record through toolbar click.

The following code snippets demonstrate how to achieve this.

CSHTML

```
@{
    List<object> toolbarItems = new List<object>();
    toolbarItems.Add("Add");
    toolbarItems.Add("Edit");
}

@Html.EJS().ContextMenu("contextmenu").Items((IEnumerable<object>)ViewBag.menuItems).Select("itemSelect").Render()
@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource).Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName").StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress").Child("SubTasks"))
).ToolbarClick("toolbarClick").Toolbar(toolbarItems).EditSettings(es=>es.AllowEditing(true).AllowAdding(true).AllowDeleting(true)).Render()
<script>

    function toolbarClick(args) {
        if (args.item.text === "Add") {
            args.cancel = true;
            var contextMenuObj =
document.getElementById("contextmenu").ej2_instances[0];
            contextMenuObj.open(60, 78);
        }
    }

    function itemSelect(args) {
        var ganttObj = document.getElementById("Gantt").ej2_instances[0];
        if (args.item.text === "Bottom") {
            ganttObj.editSettings.newRowPosition = "Bottom";
            ganttObj.openAddDialog();
        } else if (args.item.text === "Above") {
            if (ganttObj.selectedRowIndex == -1) {
                alert("Please select any row");
            } else {
                ganttObj.editSettings.newRowPosition = "Above";
                ganttObj.openAddDialog();
            }
        } else if (args.item.text === "Below") {
            if (ganttObj.selectedRowIndex == -1) {
                alert("Please select any row");
            } else {
                ganttObj.editSettings.newRowPosition = "Below";
                ganttObj.openAddDialog();
            }
        } else if (args.item.text === "Child") {
            if (ganttObj.selectedRowIndex == -1) {
```

```

        alert("Please select any row");
    } else {
        ganttObj.editSettings.newRowPosition = "Child";
        ganttObj.openAddDialog();
    }
} else if (args.item.text === "Top") {
    ganttObj.editSettings.newRowPosition = "Top";
    ganttObj.openAddDialog();
}
}

</script>

```

NEWROWPOSITION.CS

```

public IActionResult Index()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "Top"
    });
    menuItems.Add(new
    {
        text = "Bottom"
    });
    menuItems.Add(new
    {
        text = "Above"
    });
    menuItems.Add(new
    {
        text = "Below"
    });
    menuItems.Add(new
    {
        text = "Child"
    });
    ViewBag.menuItems = menuItems;
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}

```

Open add/edit dialog dynamically

Restriction of collapsing the records when clicking on gantt chart rows can be performed by using the [collapsing](#) event.

CSHTML

```

@Html.EJS().Gantt("Gantt").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("450px").TaskFields(ts => ts.Id("TaskId").Name("TaskName")
.StartDate("StartDate").EndDate("EndDate").Duration("Duration").Progress("Progress")
.Child("SubTasks")).EditSettings(es =>

```

```
es.AllowEditing(true).AllowAdding(true)).Collapsing("Collapsing").Render()

<script>
function Collapsing(args) {
    var gantt = document.getElementById("Gantt").ej2_instances[0];
    if (gantt.ganttChartModule.isExpandCollapseFromChart) {
        args.cancel = true;
    }
}
</script>
```

RESTRICTCOLLAPSECHARTROWS.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = GanttData.ProjectNewData();
    return View();
}
```

Grid

Getting Started with ASP.NET MVC Grid Control

This section briefly explains about how to include [ASP.NET MVC Grid](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Grid control

Now, add the Syncfusion ASP.NET MVC Grid control in **~/Views/Home/Index.cshtml** page.

CSHTML

```
@Html.EJS().Grid("Grid").Render()
```

Defining Row Data

To bind data for the Grid component, you can assign a `IEnumerable` object to the [DataSource](#) property. The list data source can also be provided as an instance of the **DataManager**.

CSHTML

```
@model List<GridSample.Controllers.OrdersDetails>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View(OrdersDetails.GetAllRecords());
    }
}

public class OrdersDetails
{
    public static List<OrdersDetails> order = new List<OrdersDetails>();
    public OrdersDetails()
    {
    }
    public OrdersDetails(int OrderID, string CustomerId, int EmployeeId, double
    Freight, bool Verified, DateTime OrderDate, string ShipCity, string
    ShipName, string ShipCountry, DateTime ShippedDate, string ShipAddress)
    {
        this.OrderID = OrderID;
        this.CustomerID = CustomerId;
        this.EmployeeID = EmployeeId;
        this.Freight = Freight;
        this.ShipCity = ShipCity;
        this.Verified = Verified;
        this.OrderDate = OrderDate;
        this.ShipName = ShipName;
        this.ShipCountry = ShipCountry;
        this.ShippedDate = ShippedDate;
        this.ShipAddress = ShipAddress;
    }
    public static List<OrdersDetails> GetAllRecords()
    {
        if (order.Count() == 0)
        {
            int code = 10000;
            for (int i = 1; i < 5; i++)
            {
                order.Add(new OrdersDetails(code + 1, "ALFKI", i + 0, 2.3 * i, false, new
                DateTime(1991, 05, 15), "Berlin", "Simons bistro", "Denmark", new
                DateTime(1996, 7, 16), "Kirchgasse 6"));
                order.Add(new OrdersDetails(code + 2, "ANATR", i + 2, 3.3 * i, true, new
                DateTime(1990, 04, 04), "Madrid", "Queen Cozinha", "Brazil", new
                DateTime(1996, 9, 11), "Avda. Azteca 123"));
                order.Add(new OrdersDetails(code + 3, "ANTON", i + 1, 4.3 * i, true, new
                DateTime(1957, 11, 30), "Cholchester", "Frankenversand", "Germany", new
                DateTime(1996, 10, 7), "Carrera 52 con Ave. Bolívar #65-98 Llano Largo"));
            }
        }
    }
}
```

```

order.Add(new OrdersDetails(code + 4, "BLONP", i + 3, 5.3 * i, false, new
DateTime(1930, 10, 22), "Marseille", "Ernst Handel", "Austria", new
DateTime(1996, 12, 30), "Magazinweg 7"));
order.Add(new OrdersDetails(code + 5, "BOLID", i + 4, 6.3 * i, true, new
DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes", "Switzerland", new
DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
code += 5;
}
}
return order;
}

public int? OrderID { get; set; }
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
public bool Verified { get; set; }
public DateTime OrderDate { get; set; }
public string ShipName { get; set; }
public string ShipCountry { get; set; }
public DateTime ShippedDate { get; set; }
public string ShipAddress { get; set; }
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Grid control will be rendered in the default web browser.

Orde...	Cust...	Empl...	Freig...	Ship...	Verifi...	Orde...	Ship...	Ship...	Ship...	Ship...
10001	ALFKI	1	2.3	Berlin	false	Wed ...	Simo...	Den...	Tue J...	Kirch...
10002	ANATR	3	3.3	Madrid	true	Wed ...	Quee...	Brazil	Wed ...	Avda...
10003	ANT...	2	4.3	Cholc...	true	Sat N...	Frank...	Germ...	Mon ...	Carre...
10004	BLONP	4	5.3	Mars...	false	Wed ...	Ernst ...	Austria	Mon ...	Mag...
10005	BOLID	5	6.3	Tsaw...	true	Wed ...	Hana...	Switz...	Wed ...	1029 ...
10006	ALFKI	2	4.6	Berlin	false	Wed ...	Simo...	Den...	Tue J...	Kirch...
10007	ANATR	4	6.6	Madrid	true	Wed ...	Quee...	Brazil	Wed ...	Avda...
10008	ANT...	3	8.6	Cholc...	true	Sat N...	Frank...	Germ...	Mon ...	Carre...
10009	BLONP	5	10.6	Mars...	false	Wed ...	Ernst ...	Austria	Mon ...	Mag...
10010	BOLID	6	12.6	Tsaw...	true	Wed ...	Hana...	Switz...	Wed ...	1029 ...

Defining Columns

The columns are automatically generated when columns declaration is empty or undefined while initializing the grid.

The Grid has an option to define columns using [Columns](#) property.

Let's check the properties used here:

- The [Field](#) property is to map with a property name an array of JavaScript objects.
- The [HeaderText](#) property is to change the title of columns.
- The [TextAlign](#) property is to change the alignment of columns. By default, columns will be left aligned. To change columns to right align, you need to define **textAlign** as **Right**.
- Using [Format](#) property you can format number and date values to standard or custom formats. Here, you have defined it for the conversion of numeric values to currency.

CSHTML

```
@model List<GridSample.Controllers.OrdersDetails>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)Model).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).Render()
```

Order ID	Customer Name	Order Date	Ship Country
10001	ALFKI	5/15/1991	Denmark
10002	ANATR	4/4/1990	Brazil
10003	ANTON	11/30/1957	Germany
10004	BLONP	10/22/1930	Austria
10005	BOLID	2/18/1953	Switzerland
10006	ALFKI	5/15/1991	Denmark
10007	ANATR	4/4/1990	Brazil
10008	ANTON	11/30/1957	Germany
10009	BLONP	10/22/1930	Austria
10010	BOLID	2/18/1953	Switzerland

Enable Paging

The paging feature enables users to view the grid record in a paged view. It can be enabled by setting the [AllowPaging](#) property to true. Pager can be customized using the [PageSettings](#) property.

CSHTML

```
@model List<GridSample.Controllers.OrdersDetails>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)Model).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging(true).PageSettings(page => page.PageSize(5)).Render()
```

Order ID	Customer Name	Order Date	Ship Country
10001	ALFKI	5/15/1991	Denmark
10002	ANATR	4/4/1990	Brazil
10003	ANTON	11/30/1957	Germany
10004	BLONP	10/22/1930	Austria
10005	BOLID	2/18/1953	Switzerland
<div> ⏪ < 1 of 4 pages > ⏩ </div>			

Enable Sorting

The sorting feature enables you to order the records. It can be enabled by setting the [AllowSorting](#) property as true. Sorting feature can be customized using the [SortSettings](#) property.

CSHTML

```
@model List<GridSample.Controllers.OrdersDetails>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)Model).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging(true).AllowSorting(true).PageSettings(page =>
page.PageSize(5)).Render()
```

Order ID	Customer Name	Order Date	Ship Country
10001	ALFKI	5/15/1991	Denmark
10006	ALFKI	5/15/1991	Denmark
10011	ALFKI	5/15/1991	Denmark
10016	ALFKI	5/15/1991	Denmark
10002	ANATR	4/4/1990	Brazil
<div> ⏪ ⏴ 1 of 4 pages ⏵ ⏩ </div>			

Enable Filtering

The filtering feature enables you to view reduced amount of records based on filter criteria. It can be enabled by setting the [AllowFiltering](#) property as true. Filtering feature can be customized using the [FilterSettings](#) property.

CSHTML

```
@model List<GridSample.Controllers.OrdersDetails>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)Model).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging(true).AllowSorting(true).AllowFiltering(true).PageSettings(pa
ge => page.PageSize(5)).Render()
```

Order ID	Customer Name	Order Date	Ship Country
	B		
10004	BLONP	10/22/1930	Austria
10005	BOLID	2/18/1953	Switzerland
10009	BLONP	10/22/1930	Austria
10010	BOLID	2/18/1953	Switzerland
10014	BLONP	10/22/1930	Austria
<div> <div>⏪</div> <div>⏩</div> <div>1 of 2 pages</div> <div>⏪</div> <div>⏩</div> </div> <div>Customer Name: B</div>			

Enable Grouping

The grouping feature enables users to view the grid record in a grouped view. It can be enabled by setting the [AllowGrouping](#) property to true. Grouping feature can be customized using the [GroupSettings](#) property.

CSHTML

```
@model List<GridSample.Controllers.OrdersDetails>
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)Model).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging(true).AllowSorting(true).AllowFiltering(true).AllowGrouping(t
rue).PageSettings(page => page.PageSize(5)).Render()
```

Drag a column header here to group its column			
Order ID	Customer Name	Order Date	Ship Country
10001	ALFKI	5/15/1991	Denmark
10002	ANATR	4/4/1990	Brazil
10003	ANTON	11/30/1957	Germany
10004	BLONP	10/22/1930	Austria
10005	BOLID	2/18/1953	Switzerland
<div> ⏪ < 1 of 4 pages > ⏩ </div>			

Note: [View Sample in GitHub.](#)

See also

- [Create an MVC Grid app](#)
- [How to render EJ2-Grid as HTML in ASP.NET MVC Grid](#)
- [How to render Grid in partial view in ASP.NET MVC Grid](#)

Data Binding

The Grid uses DataManager, which supports both RESTful JSON data services binding and local JavaScript object array binding. The [DataSource](#) property can be assigned either with the instance of DataManager or JavaScript object array collection.

It supports the following kinds of data binding method:

- List binding
- DataTable binding
- Remote data

Sending additional parameters to the server

To add a custom parameter to the data request, use the addParams method of **Query** class. Assign the **Query** object with additional parameters to the grid [Query](#) property.

CSHTML

```
@Html.EJS().Grid("Params").DataSource(dataManger =>
{
    dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/Orders").CrossDomain(true).Adaptor("ODataV4Adaptor");
}).Query("new ej.data.Query().addParams('ej2Grid', 'true')").Columns(col =>
{
```

```
col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width("160").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Freight").HeaderText("Freight").Width("150").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().Render()
```

PARAMS.CS

```
public IActionResult Index()
{

    return View();
}
```

Note: The parameters added using the [Query](#) property will be sent along with the data request for every grid action.

Handling HTTP error

During server interaction from the grid, some server-side exceptions may occur, and you can acquire those error messages or exception details in client-side using the [ActionFailure](#) event.

The argument passed to the [ActionFailure](#) event contains the error details returned from the server.

CSHTML

```
@Html.EJS().Grid("check").DataSource(dataManger =>
{
    dataManger.Url("http://some.com/invalidUrl");
}).Columns(col =>
{
    col.Field("ProductID").HeaderText("Product
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ProductName").HeaderText("Product Name").Width("150").Add();
    col.Field("UnitPrice").HeaderText("Supplier
ID").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("UnitsInStock").HeaderText("QuantityPerUnit").Width("120").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Discontinued").HeaderText("Discontinued").Width("140").TextAlign(
Syncfusion.EJ2.Grids.TextAlign.Center).Type("boolean").DisplayAsCheckBox(tru
e).Add();
}).ActionFailure("actionFailure").Render()
<script>
    function actionFailure(args) {
        var span = document.createElement('span');
        this.element.parentNode.insertBefore(span, this.element);
        span.style.color = '#FF0000'
        span.innerHTML = 'Server exception: 404 Not found';
    }
```

```
</script>
```

HTTPERROR.CS

```
public IActionResult Index()
{
    return View();
}
```

Note: The [ActionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the grid actions.

Binding with ajax

You can use Grid [DataSource](#) property to bind the datasource to Grid from external ajax request. In the below code we have fetched the datasource from the server with the help of ajax request and provided that to [DataSource](#) property by using `onSuccess` event of the ajax.

```
`javascript
<script type="text/javascript">
var grid = document.getElementById('Grid').ej2_instances[0]; // Grid instance
var ajax = new ej.base.Ajax('/Home/UrlDatasource', 'GET');
ajax.send();
ajax.onSuccess = function (data) {
    grid.dataSource = JSON.parse(data).result;
};
</script>
```

Note: * If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

Troubleshoot: Grid render rows without data

In ASP.NET Core, by default the JSON results are returned in camelCase format. So grid field names are also changed in camelCase.

To avoid this problem, you need to add **DefaultContractResolver** in **Startup.cs** file.

```
`javascript
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().AddJsonOptions(options =>
    {
        options.SerializerSettings.ContractResolver = new
        Newtonsoft.Json.Serialization.DefaultContractResolver();
    });
}
```

```
});
}
,
```

See Also

- [JSON binding to Grid and monitoring with knockout.js](#)
- [How to bind the Grid data using Dictionary collection in ASP.NET MVC Grid](#)

Data Annotation in Grid Component

Data Annotation helps you define rules for the model class to perform data validation and display suitable messages to end users as validation message in the edit form.

The Data Annotation can be enabled by referencing the **System.ComponentModel.DataAnnotations** namespace which maps data annotation to the corresponding DataGrid Column property.

Note: The DataGrid Property has more priority than the Data Annotation. For Instance, if the DisplayName Attribute is set to a Field in the DataGrid model class and the HeaderText is set to the same DataGrid column property, the value of the HeaderText property will be considered and shown in the DataGrid header.

CSHTML

```
@using DataAnnotation.Controllers;
@(Html.EJS().Grid<HomeController.Orders>("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Width("150").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

    col.Field("Freight").HeaderText("Freight").EditType("numericedit").Width("160").Format("N2").Add();
    col.Field("ShipCountry").EditType("dropdownedit").HeaderText("Ship Country").Width("140").Add();
}).AllowPaging().PageSettings(page => page.PageCount(2)).EditSettings(edit => { edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true);
}).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel"
}).Render())
```

ANNOTATION.CS

```
namespace DataAnnotation.Controllers
{
    public class HomeController : Controller
    {
        public static List<Orders> order = new List<Orders>();
        public ActionResult Index()
        {
            ViewBag.DataSource = Orders.getAllRecords().ToList();
            ViewBag.Type = typeof(Orders);
            return View();
        }
        public class Orders
```

```

    {
        public Orders(long OrderId, string Customerid, int EmployeeId,
double Freight, string ShipCity)
        {
            this.OrderID = OrderId;
            this.CustomerID = Customerid;
            this.EmployeeID = EmployeeId;
            this.Freight = Freight;
            this.ShipCity = ShipCity;
        }
        public static List<Orders> getAllRecords()
        {
            if (order.Count == 0)
            {
                int code = 10000;
                for (int i = 1; i < 2; i++)
                {
                    order.Add(new Orders(code + 1, "ALFKI", i + 0, 2.3 *
i, "Berlin"));
                    order.Add(new Orders(code + 2, "ANATR", i + 2, 3.3 *
i, "Madrid"));
                    order.Add(new Orders(code + 3, "ANTON", i + 1, 4.3 *
i, "Cholchester"));
                    order.Add(new Orders(code + 4, "BLONP", i + 3, 5.3 *
i, "Marseille"));
                    order.Add(new Orders(code + 5, "BOLID", i + 4, 6.3 *
i, "Tsawassen"));
                    code += 5;
                }
            }
            return order;
        }
        [Key]
        [DisplayName = "Order ID"]
        [Required(ErrorMessage = "Order ID is required")]
        public long OrderID { get; set; }
        [DisplayName = "Customer ID"]
        [Required(ErrorMessage = "Customer ID is required")]
        [StringLength(8, MinimumLength = 3, ErrorMessage = "Customer ID
length should between 3 and 8")]
        public string CustomerID { get; set; }
        [DisplayName = "Employee ID"]
        [Range(1, 10, ErrorMessage = "Employee ID should be between 1
and 10")]
        public int EmployeeID { get; set; }
        [DisplayFormat(DataFormatString = "c2")]
        [Range(1, 1000, ErrorMessage = "Freight should be between 1 and
1000")]
        public double Freight { get; set; }
        [DisplayName = "Ship City"]
        [Editable(false)]
        public string ShipCity { get; set; }
    }
}

```


Immutable Mode

The immutable mode optimizes the Grid re-rendering performance by using the object reference and [deep compare](#) concept. When performing the Grid actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

Note: * This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.EnableImmutableMode(true).AllowPaging(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.
n.Right).Add();
    col.Field("ProductName").HeaderText("Product Name").Width("160").Add();
    col.Field("ProductID").HeaderText("Product
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("120").Add();
    col.Field("CustomerName").HeaderText("Customer
Name").Width("160").Add();
}).Render()
```

IMMUTABLE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Hierarchy Grid
- Column reorder
- Virtual scroll
- Infinite scroll
- Grouping

Performance tips for ASP.NET MVC Grid control

This article is a comprehensive guide on improving the loading performance of the ASP.NET MVC Grid, especially when dealing with large datasets along with large number of columns. It provides valuable insights into the steps that need to be followed to bind a large data source without experiencing any performance degradations. By offering detailed explanations and actionable tips, this resource aims to

empower readers with the knowledge and best practices necessary to optimize the performance of the ASP.NET MVC Grid during data binding, ensuring a smooth and efficient user experience.

How to improve loading performance by binding large dataset

As you all know, a grid is made up of rows and columns. For instance, when you bind 10 rows and 10 columns, it means 100 elements will be rendered in the DOM (Document Object Model). So, it is recommended to render only a limited number of rows and columns to guarantee the best loading performance for the control.

Optimizing performance with paging

To boost the performance efficiency of your application, especially when dealing with large datasets, it is advised to implement paging. [Paging](#) allows you to display grid data in segmented pages, facilitating easier navigation through substantial datasets. This feature proves particularly beneficial in enhancing the overall performance of your application. For more information on implementing paging, you can refer to the [documentation](#) section dedicated to this feature.

Optimizing performance with row virtualization or infinite scrolling

To enhance your application's efficiency, especially when dealing with substantial datasets, it is recommended to either using [virtualization](#) or [infinite scrolling](#). Implementing these techniques can significantly reduce the load on your application and elevate its overall performance.

1. **Virtualization:** The Virtual scrolling feature in the ASP.NET MVC Grid enables the efficient handling and display of large volumes of data without compromising performance. This approach optimizes the rendering process by loading only the visible rows within the Grid viewport, rather than rendering the entire dataset simultaneously. For more information on implementing row virtualization , you can refer to the [documentation](#) section dedicated to this feature.
2. **Infinite scrolling:** The Infinite Scrolling feature in the ASP.NET MVC Grid is a powerful tool for seamlessly handling extensive data sets without compromising grid performance. It operates on a "load-on-demand" concept, ensuring that data is fetched only when needed. In the default infinite scrolling mode, a new block of data is loaded each time the scrollbar reaches the end of the vertical scroller. For more information on implementing infinite scrolling , you can refer to the [documentation](#) section dedicated to this feature.

Optimizing performance with column virtualization in large no of columns

[Column virtualization](#) feature in the ASP.NET MVC Grid that allows you to optimize the rendering of columns by displaying only the columns that are currently within the viewport. It allows horizontal scrolling to view additional columns. This feature is particularly useful when dealing with grids that have a large number of columns, as it helps to improve the performance and reduce the initial loading time.

It is possible to enable both row and column virtualization. This feature allows for efficient handling of large datasets by dynamically loading only the visible rows and columns, optimizing performance and enhancing the overall responsiveness of the grid. For more information on implementing column virtualization , you can refer to the [documentation](#) section dedicated to this feature.

How to overcome browser height limitation in virtual scrolling

[Documentation link](#)

How to improve loading performance by binding large data by showing custom text or element
When integrating image or template elements into a column, it's recommended to utilize the [Column Template](#) feature rather than customizing the data through [rowDataBound](#) or [queryCellInfo](#) events.

These events are triggered for each row and cell rendering, introducing delays in the control's rendering process. Moreover, rendering custom elements using these events may result in the persistence of rendered elements, potentially causing longer rendering times over time. By opting for the column template feature, you can efficiently meet this requirement without experiencing rendering delays and ensure a more streamlined rendering process.

[How to improve loading performance by referring individual script and CSS](#)

To improve the performance of Syncfusion Grid control during the initial render as well as certain actions, suggested you to download the specific control scripts using CRG (Custom Resource Generator) to speed up the project. By default, the ej2.min.js script file contains all the Syncfusion control scripts. So, it will take some time to load the scripts to the project. Using [CRG](#), you can select the controls which you want to use, and the modules for those controls, then you can download the scripts and CSS for the selected controls and use them as per your need.

[CRG website link](#)

So to improve the performance of grid during the initial rendering, suggested you to refer individual script and CSS.

[How to update cell values without frequent server calls](#)

Efficiently update cell values without the need for frequent server calls, especially beneficial for live update scenarios. Even when the data is initially bound from the server, performing edit operations can be done without triggering a database refresh. Utilize the [setCellValue](#) method to update the Grid without affecting the database and only refresh the UI.

[How to optimize server-side data operations with adaptors](#)

The ASP.NET MVC Grid provides support for various adaptors (OData, ODataV4, WebAPI, URL, etc.) to facilitate server-side data operations and CRUD functionalities. By leveraging these adaptors along with the **DataManager** control, you can seamlessly bind remote data sources to the grid and execute actions. During data operations like filtering, sorting, and paging, the corresponding action queries are generated as per the adaptor's requirements. It is crucial to handle these actions on the application end and return the processed data back to the grid. Refer to the documentation for comprehensive details. It's worth noting that for efficient data processing, the suggested order for returning processed data to the grid is as follows

- Filtering
- Sorting
- Aggregates
- Paging
- Grouping

[How to avoid MaxJsonLength error while passing large amount of records](#)

The ASP.NET MVC Grid control is client-server based. So, we send the data as JSON object between client and server. The reported issue occurs due to the serialization of the large-sized JSON object. We need to increase the maximum length for serializing the large-sized JSON object. You have to alter the [MaxJsonLength](#) property on your web.config file or in the place of deserialization.

Solution: 1

```
`csharp
<configuration>
```

```
<system.web.extensions>
<scripting>
<webServices>
<jsonSerialization maxJsonLength="25000000"/>
</webServices>
</scripting>
</system.web.extensions>
</configuration>
```

Solution : 2

```
`csharp
var serializer = new JavaScriptSerializer { MaxJsonLength = Int32.MaxValue };
`
```

Microsoft excel limitation while exporting millions of records to excel file format

By default, Microsoft Excel supports only 1,048,576 records in an excel sheet. Hence it is not possible to export millions of records to excel. You can refer the [documentation](#) link for more details on Microsoft excel specifications and limits. So suggest to export the data in CSV (Comma-Separated Values) or other formats that can handle large datasets more efficiently than Excel.

Columns in ASP.NET MVC Grid Component

The column definitions are used as the [DataSource](#) schema in the Grid. This plays a vital role in rendering column values in the required format.

The grid operations such as sorting, filtering and grouping etc. are performed based on column definitions. The [Field](#) property of the [Columns](#) is necessary to map the data source values in Grid columns.

Note: 1. If the column [Field](#) is not specified in the dataSource, the column values will be empty.

 2. If the [Field](#) name contains “dot” operator, it is considered as complex binding.

Column types

Column type can be specified using the [Type](#) property of [Columns](#). It specifies the type of data the column binds.

If the [Format](#) is defined for a column, the column uses [Type](#) to select the appropriate format option **number** or **date**.

Grid column supports the following types:

- string
- number
- boolean
- date
- datetime

Note: If the [Type](#) is not defined, it will be determined from the first record of the [DataSource](#).

 In case if the first record of the [DataSource](#) is null/blank value for a column then it is necessary to define the [Type](#) for that column.

ValueAccessor

The **valueAccessor** is used to access/manipulate the value of display data. You can achieve custom value formatting by using the valueAccessor.

CSHTML

```
@{
    var valueAccessor = "valueAccessorFn";
}
@Html.EJS().Grid("ValueAccessor").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShipCity").HeaderText("Ship City").ValueAccessor(valueAccessor).Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("130").Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Format("C2").Add();
}).AllowPaging().Render()
<script>
    function valueAccessorFn(field, data, column){
        var value = data[field]
        if (data['ShipCountry'] !== undefined) {
            value = value + ' - ' + data['ShipCountry'];
        }
        return value;
    }
</script>
```

VALUEACCESSOR.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Format

To format cell values based on specific culture, use the [Format](#) property of [Columns](#). The grid uses Internalization library to format number and date values.

CSHTML

```
@Html.EJS().Grid("Format").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
```

```
{
    col.Field("OrderID").HeaderText("Order
ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("130").Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.Te
xtAlign.Right).Width("120").Format("C2").Add();
}).AllowPaging().Render();
```

FORMAT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: By default, the number and date values are formatted in **en-US** locale.

Number formatting

The number or integer values can be formatted using the below format strings.

Format	Description	Remarks
----- ----- -----		
N	Denotes numeric type.	The numeric format is followed by integer value as N2, N3. etc which denotes the number of precision to be allowed.
C	Denotes currency type.	The currency format is followed by integer value as C2, C3. etc which denotes the number of precision to be allowed.
P	Denotes percentage type	The percentage format expects the input value to be in the range of 0 to 1. For example the cell value 0.2 is formatted as 20% . The percentage format is followed by integer value as P2, P3. etc which denotes the number of precision to be allowed.

Date formatting

You can format date values either using built-in date format string or custom format string.

For built-in date format you can specify [Format](#) property as string(Example: `yMd`).

You can also use custom format string to format the date values. Some of the custom formats and the formatted date values are given in the below table.

Format	Formatted value
----- -----	
{ type:'date', format:'dd/MM/yyyy' }	04/07/1996
{ type:'date', format:'dd.MM.yyyy' }	04.07.1996
{ type:'date', skeleton:'short' }	7/4/96

| { type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM |

| { type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM |

CSHTML

```
@Html.EJS().Grid("Format").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("130").Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.Te
xtAlign.Right).Width("120").Format("C2").Add();

    col.Field("Price").HeaderText("Price").TextAlign(Syncfusion.EJ2.Grids.TextAl
ign.Right).Width("120").Format("P2").Add();
}).AllowPaging().Render()
```

FORMAT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Render boolean value as checkbox

To render boolean values as checkbox in columns, you need to set [DisplayAsCheckBox](#) property as **true**.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.dataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").ValidationRules(new { required = "true"
}).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Width("150").ValidationRules(new { required = "true", minLength = 3
}).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();

    col.Field("Verified").HeaderText("Verified").DisplayAsCheckBox(true).Width("
150").Add();
}).AllowPaging().Render()
```

BOOLEANASCHECKBOX.CS

```
public IActionResult Index()
{
    var orders = OrdersDetails.GetAllRecords();
    ViewBag.DataSource = orders;
    return View();
}
```

Visibility

You can hide any particular column in Grid before rendering by defining [Visible](#) property as false. In the below sample **ShipCity** column is defined as visible false.

CSHTML

```
@Html.EJS().Grid("Visibility").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShippedDate").HeaderText("Shipped Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCity").Visible(false).HeaderText("Ship City").Width("150").Add();
}).AllowPaging().Render()
```

VISIBILITY.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Lock columns

You can lock columns by using [LockColumn](#) property. The locked columns will be moved to the first position. Also you cannot reorder its position.

In the below example, Ship City column is locked and its reordering functionality is disabled.

CSHTML

```
@Html.EJS().Grid("Reorder").DataSource((IEnumerable<object>)ViewBag.dataSource).AllowReordering().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
    col.Field("ShipCity").Width("130").LockColumn(true).Add();
    col.Field("ShipName").Width("150").Add();
    col.Field("ShipPostalCode").Width("125").Add();
    col.Field("ShipRegion").Width("120").Add();
}).Render()
```


LOCK.CS

```
public IActionResult Index()
{
    var order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = order;
    return View();
}
```

Controlling Grid actions

You can enable or disable grid action for a particular column by setting the [AllowFiltering](#), [AllowGrouping](#), [AllowEditing](#), [AllowReordering](#), and [AllowSorting](#) properties.

CSHTML

```
@Html.EJS().Grid("ControlGridAction").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).AllowGrouping(false).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").AllowFiltering(false).AllowEditing(false).AllowReordering(false).Add();
    col.Field("OrderDate").HeaderText("Order Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("130").Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").AllowSorting(false).Format("C2").Add();
})
.AllowPaging().AllowGrouping().AllowFiltering().AllowSorting().AllowReordering().EditSettings(edit => {
    edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true);
}).Render()
```

CONTROLLINGACTIONS.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Show or hide columns by external button

You can show or hide grid columns dynamically using external buttons by invoking the **showColumns** or **hideColumns** method.

CSHTML

```
@Html.EJS().Button("Show").Content("Show").IsPrimary(true).Render()
@Html.EJS().Button("Hide").Content("Hide").IsPrimary(true).Render()
@Html.EJS().Grid("ShowHide").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
```

```

{
    col.Field("OrderID").HeaderText("Order
ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("130").Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.Te
xtAlign.Right).Width("120").Format("C2").Add();
}).AllowPaging().Render()
<script>
    document.getElementById("Show").addEventListener("click", () => {
        var grid = document.getElementById("ShowHide").ej2_instances[0];
        grid.showColumns(['Customer Name', 'Order Date']); //show by
HeaderText
    });
    document.getElementById("Hide").addEventListener("click", () => {
        var grid = document.getElementById("ShowHide").ej2_instances[0];
        grid.hideColumns(['Customer Name', 'Order Date']); //hide by
HeaderText
    })
</script>

```

SHOWHIDE.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Customize column styles

You can customize the appearance of the header and content of a particular column using the [customAttributes](#) property.

To customize the grid column, follow the given steps:

Step 1: Create a CSS class with custom style to override the default style for rowcell and headercell.

```

`css
.e-grid .e-rowcell.customcss{
background-color: #ecedee;
color: 'red';
font-family: 'Bell MT';
font-size: 20px;
}
.e-grid .e-headercell.customcss{
background-color: #2382c3;

```

```
color: white;
font-family: 'Bell MT';
font-size: 20px;
}
`
```

Step 2: Add the custom CSS class to the specified column by using the [customAttributes](#) property.

```
`typescript
```

```
col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).CustomAttributes(new { @class = "customcss" }).Add();
`
```

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).CustomAttributes(new { @class =
"customcss" }).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Render()
<style>
    .e-grid .e-rowcell.customcss {
        background-color: #ecedee;
        color: red;
        font-family: 'Bell MT';
        font-size: 20px;
    }
    .e-grid .e-headercell.customcss {
        background-color: #2382c3;
        color: white;
        font-family: 'Bell MT';
        font-size: 20px;
    }
</style>
```

CUSTOM-COLUMN-STYLE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Display custom tooltip for columns

To display a custom ToolTip (**EJ2 Tooltip**), you can render the Grid control inside the Tooltip component and set the target as ".e-rowcell". The tooltip is displayed when hovering the grid cells.

Change the tooltip content for the grid cells by using the following code in the (**beforeRender**) event.

`typescript

```
function beforeRender(args) {
    // event triggered before render the tooltip on target element.
    var tooltip = document.getElementById("Tooltip").ej2_instances[0]
    tooltip.content = args.target.closest("td").innerText;
}
```

CSHTML

```
@Html.EJS().Tooltip("Tooltip").Target(".e-rowcell").ContentTemplate(@<div>

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").Width("120").Add();
}).Render()
</div>).BeforeRender("beforeRender").Render()
<script>
    function beforeRender(args) {
        // event triggered before render the tooltip on target element.
        var tooltip = document.getElementById("Tooltip").ej2_instances[0]
        tooltip.content = args.target.closest("td").innerText;
    }
</script>
```

CUSTOM-TOOLTIP.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Align the text of Grid content and header

For aligning the text of Grid content and header part, use [TextAlign](#) and [HeaderTextAlign](#) properties.

Grid column supports the following alignments:

- Left
- Right
- Center
- Justify

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).HeaderText
Align(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Width("150").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Left).HeaderTex
tAlign(Syncfusion.EJ2.Grids.TextAlign.Left).Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Center).HeaderT
extAlign(Syncfusion.EJ2.Grids.TextAlign.Center).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Justify).HeaderTextAlign(Syncfusion.EJ2.Gr
ids.TextAlign.Justify).Add();
}).Render()
```

ALIGNMENT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

How to prevent checkbox in the blank row

By default, cells in the grid will be blank if the corresponding column values in the data source are null or undefined. The grid also has the option to prevent the rendering of checkboxes in such cases, even if the [DisplayAsCheckBox](#) property is set to true for that column, by using the [RowDataBound](#) event of the Grid.

In the following sample, the [RowDataBound](#) event of the Grid is used to set the innerHTML of the checkbox element to empty.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
```

```

        col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
        col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

        col.Field("Verified").HeaderText("Verified").Width("120").DisplayAsCheckBox(true).Add();
    }).RowDataBound("rowBound").Render()
<script>
    function rowBound(args) {
        let grid = document.getElementById('Grid').ej2_instances[0];
        let count = 0;
        let keys = Object.keys(args.data);
        for (let i = 0; i < keys.length; i++) {
            if (args.data[keys[i]] == null || args.data[keys[i]] == '' || args.data[keys[i]] == undefined) {
                count++;
            }
        }
        if (count == keys.length) {
            for (let i = 0; i < grid.columns.length; i++) {
                if (grid.columns[i].displayAsCheckBox) {
                    args.row.children[i].innerHTML = '';
                }
            }
        }
    }
</script>

```

BLANK-ROW.CS

```

public IActionResult Index()
{
    var orders = OrdersDetails.GetAllRecords();
    ViewBag.DataSource = orders;
    return View();
}

```

See Also

- [Group Column by Format](#)
- [How to set complex column as Foreignkey column](#)
- [Complex Data Binding with list of Array Of Objects](#)
- [How to add primaryKey after column rendered in ASP.NET MVC Grid](#)
- [Drag and drop between two Grids in ASP.NET MVC Grid](#)
- [How can I put a Sparkline in a child grid that is on a Grid with hierarchy in ASP.NET MVC Grid](#)
- [How can I enable or disable a menu option that is inside a template in ASP.NET MVC Grid](#)
- [How to change the data source or columns dynamically](#)

Row in ASP.NET MVC Grid Control

The row represents record details fetched from data source.

Row customization

Using event

You can customize the appearance of a row by using the [RowDataBound](#) event. The [RowDataBound](#) event triggers for every row. In the event handler, you can get the **RowDataBoundEventArgs** that contains details of the row.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).RowDataBound("rowBound").AllowSelection(false).PageSettings(p => {
p.PageSize(6); }).EnableHover(false).Render()
<script>
    function rowBound(args) {
        if (args.data['Freight'] < 10) {
            args.row.classList.add('below-10');
        } else if (args.data['Freight'] < 80) {
            args.row.classList.add('below-80');
        } else {
            args.row.classList.add('above-80');
        }
    }
</script>
<style>
    .below-10 {
        background-color:aqua;
    }
    .below-80 {
        background-color: aquamarine;
    }
    .above-80 {
        background-color: orange;
    }
</style>
```

CUSTOM-ROWS.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Styling alternate rows

You can change the grid's alternative rows' background color by overriding the **.e-altrow** class.

```
`css
.e-grid .e-altrow {
background-color: #fafafa;
}
```

Refer to the following example.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).Render()
<style>
    .e-grid .e-altrow {
        background-color: #fafafa;
    }
</style>
```

STYLE-ALT-ROW.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Using CSS customize selected row

The background color of the selected row can be changed by overriding the following CSS style.

```
`css
.e-grid td.e-active {
background-color: #f9920b;
}
```

This is demonstrated in the following sample:

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).Render()
<style>
    .e-grid td.e-active {
        background-color: #f9920b;
    }
</style>
```

SELECTED-ROW.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Adding a new row programmatically

The Grid can add a new row between the existing rows using the `addRecord` method of the Grid.

This is demonstrated in the following sample:

CSHTML

```
@Html.EJS().Button("add").Content("Add Row").IsPrimary(true).Render()
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).IsPrimaryK
ey(true).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("130").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("120").Add();
}).EditSettings(edit => {
    edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Render()
<script>
    document.getElementById('add').onclick = function () {
```

```

        document.getElementById('Grid').ej2_instances[0].addRecord(
            { OrderID: 3232, CustomerID: 'ALKIT', ShipCity: 'London', Freight:
40, ShipName: 'Que Delícia'}, 2);
    }
</script>

```

ADD-ROW.CS

```

public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}

```

Note: When working with remote data, it is impossible to add a new row between the existing rows.

How to get the row information when hovering over the cell

It is possible to get the row information when hovering over the specific cell. This can be achieved by using the [RowDataBound](#) event and `getRowInfo` method of the Grid.

In the following sample, the `mouseover` event is bound to a grid row in the `RowDataBound` event, and when hovering over the specific cell, using the `getRowInfo` method, row information will be retrieved and displayed in the console.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.RowDataBound("rowDataBound").EditSettings(edit => {
    edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true);
}).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).Render()
<script>
    function rowDataBound(args) {
        var gridElement = document.getElementById('Grid').ej2_instances[0];
        args.row.addEventListener('mouseover', function (e) {
            console.log(gridElement.getRowInfo(e.target))
        })
    }
</script>

```

ROW-INFO.CS

```

public IActionResult Index()

```

```
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

See Also

- [How to set textbox and Grid in one row in ASP.NET MVC Grid](#)

Cell

Cell customization

The appearance of cells can be customized by using the [QueryCellInfo](#) event. The [QueryCellInfo](#) event triggers for every cell. In that event handler, you can get **QueryCellInfoEventArgs** that contains the details of the cell.

CSHTML

```
@Html.EJS().Grid("CellCustomize").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().QueryCellInfo("customiseCell").Render()
<style>
    .below-5 {
        background-color: orangered;
    }
    .below-8 {
        background-color: yellow;
    }
    .above-8 {
        background-color: greenyellow;
    }
</style>
<script>
    function customiseCell(args) {
        if (args.column.field === 'Freight') {
            if (args.data['Freight'] < 5) {
                args.cell.classList.add('below-5');
            } else if (args.data['Freight'] < 8) {
                args.cell.classList.add('below-8');
            } else {
                args.cell.classList.add('above-8');
            }
        }
    }
}
```

```
</script>
```

CUSTOMIZE.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Custom attributes

You can customize the grid cells by adding a CSS class to the [customAttribute](#) property of the column.

```
`CSS
```

```
.e-attr {
background: #d7f0f4;
}
`
```

In the below example, we have customized the cells of **OrderID** and **ShipCity** columns.

CSHTML

```
@Html.EJS().Grid("CellCustomize").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").CustomAttributes(new { @class = "customcss" })
    .TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
    col.Field("ShipCity").HeaderText("Ship City").CustomAttributes(new { @class = "e-attr" })
    .Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd")
    .Add();
}).AllowPaging().Render()
<style>
    .e-attr {
        background: #d7f0f4;
    }
</style>
```

CUSTOMSTYLE.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Grid lines

The [GridLines](#) have option to display cell border and it can be defined by the [GridLines](#) property.

The available modes of grid lines are:

- | Modes | Actions |
- |-----|-----|
- | Both | Displays both the horizontal and vertical grid lines. |
- | None | No grid lines are displayed. |
- | Horizontal | Displays the horizontal grid lines only. |
- | Vertical | Displays the vertical grid lines only. |
- | Default | Displays grid lines based on the theme. |

CSHTML

```
@Html.EJS().Grid("GridLine").DataSource((IEnumerable<object>)ViewBag.dataSource).GridLines(Syncfusion.EJ2.Grids.GridLine.Both).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("C2").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().PageSettings(page => { page.PageCount(5);}).Render()
```

GRIDLINES.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: By default, the grid renders with **Default** mode.

Editing in ASP.NET MVC Grid Component

The Grid component has options to dynamically insert, delete and update records. Editing feature requires a primary key column for CRUD operations. To define the primary key, set [IsPrimaryKey](#) property of [Column](#) to **true** in particular column.

You can start the edit action either by double clicking the particular row or by selecting the required row and click on **Edit** button in the toolbar. Similarly, you can add a new record to grid either by clicking on **Add** button in the toolbar or on an external button which is bound to invoke the [addRecord](#) method of the grid, **Save** and **Cancel** while in edit mode is possible using respective toolbar icon in grid.

Deletion of the record is possible by selecting the required row and click on **Delete** button in the toolbar.

CSHTML

```
@Html.EJS().Grid("NormalEdit").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Width("120").ValidationRules(new { required = "true"}).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").ValidationRules(new { required = "true", minLength=3 }).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().PageSettings(page => page.PageCount(2)).EditSettings(edit => {
    edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Normal);
}).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel" }).Render()
```

EDIT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: If [IsIdentity](#) is enabled, then it will be considered as a read-only column when editing and adding a record.

 You can disable editing for a particular column, by specifying [AllowEditing](#) to **false**.

Toolbar with edit option

The grid toolbar has the [built-in items](#) to execute Editing actions. You can define this by using the [Toolbar](#) property.

CSHTML

```
@Html.EJS().Grid("EditToolbar").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).ValidationRules(new { required = "true"}).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").ValidationRules(new { required = "true", minLength = 3 }).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}
```

```
col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel" }).Render()
```

EDITTOOLBAR.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Disable editing for particular column

You can disable editing for particular columns by using the [AllowEditing](#) property of [Column](#).

In the following demo, editing is disabled for the **CustomerID** column.

CSHTML

```
@Html.EJS().Grid("InlineEditing").DataSource((IEnumerable<object>)ViewBag.datasources).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).ValidationRules(new { required = "true" }).Add();
    col.Field("CustomerID").HeaderText("Customer Name").AllowEditing(false).Width("150").ValidationRules(new { required = "true", minLength = 3 }).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel" }).Render()
```

DISABLEEDITFORCOLUMN.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Disable editing for a particular row or cell

You can disable the editing for a particular row by using the [ActionBegin](#) event of Grid.

In the below demo, the rows which are having the value for **ShipCountry** column as Denmark is prevented from editing.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width(30).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width(30).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width(30).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width(30).Add();
}).EditSettings(edit => edit.AllowEditing(true)).Toolbar(new List<string>()
{ "Edit", "Cancel", "Update" }).ActionBegin("actionBegin").Render()
<script>
    function actionBegin(args) {
        if (args.requestType === "beginEdit") {
            if (args.rowData.ShipCountry === "Denmark") {
                args.cancel = true;
            }
        }
    }
</script>
```

DISABLE-EDIT.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

For batch mode of editing, you can use [CellEdit](#) event of Grid. In the below demo, the cells which are having the value as Denmark is prevented from editing.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width(30).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width(30).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width(30).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width(30).Add();
}).EditSettings(edit =>
edit.AllowEditing(true).Mode(Syncfusion.EJ2.Grids.EditMode.Batch)).Toolbar(n
ew List<string>() { "Edit", "Cancel", "Update"
}).CellEdit("cellEdit").Render()
<script>
    function cellEdit(args) {
        if (args.value === "Denmark") {
            args.cancel = true;
        }
    }
}
```



```
</script>
```

DISABLE-EDIT-BATCH.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Editing template column

You can edit the template column value by defining the [Field](#) for that particular column.

In the below demo, the **ShipCountry** column is rendered with the template.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").Template("#template").EditType("dropdownedit").Add();
}).AllowPaging().EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel" }).Render()
<script id="template" type="text/x-template">
    <a href="#">${ShipCountry}</a>
</script>
```

EDIT-TEMP.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Troubleshoot editing works only for first row

The Editing functionalities can be performed based upon the primary key value of the selected row. If [IsPrimaryKey](#) is not defined in the grid, then edit or delete action take places the first row.

How to make a Grid column always editable

Make the Grid column always editable using the column template feature of the Grid.

In the following example, the textbox is rendered in the Freight column using a column template. The keyup event for the Grid is bound using the [created](#) event of the Grid, and the edited changes are saved in the data source using the `updateRow` method of the Grid.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Created("created").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).IsPrimaryK
ey(true).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
    col.Field("Freight").HeaderText("Receipt
Amount").Width("120").Template("#template").Add();
}).Render()
<script id="template" type="text/x-template">
    <input id='${OrderID}' value='${Freight}' class='custemp' type='text'
style='width: 100%'>
</script>
<script>
    function created(e) {

document.getElementById('Grid').ej2_instances[0].element.addEventListener('k
eyup', function (e) { // Bind the keyup event for the grid.
    if (e.target.classList.contains('custemp')) { // Based on this
condition, you can find whether the target is an input element or not.
        var row = ej.grids.parentsUntil(e.target, 'e-row');
        var rowIndex = row.rowIndex; // Get the row index.
        var uid = row.getAttribute('data-uid');
        var rowData =
document.getElementById('Grid').ej2_instances[0].getRowObjectFromUID(uid).da
ta; // Get the row data.
        rowData.Freight = e.target.value; // Update the new value
for the corresponding column.

document.getElementById('Grid').ej2_instances[0].updateRow(rowIndex,
rowData); // Update the modified value in the row data.
    }
});
}
</script>
```

COLUMN-EDIT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

See Also

- [Update column value based on other column values in ASP.NET MVC Grid](#)
- [Populate field depending the value selected from another field in ASP.NET MVC Grid](#)
- [How to apply two different validation in same column in ASP.NET MVC Grid](#)
- [Using Uploader and Textarea in the Grid when edit record in ASP.NET MVC Grid](#)

Sorting

Sorting enables you to sort data in the **Ascending** or **Descending** order. To sort a column, click the column header.

To enable sorting in the Grid, set the [AllowSorting](#) to true. Sorting options can be configured through the [SortSettings](#).

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>) ViewBag.datasource)
.AllowSorting().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging().PageSettings(page => page.PageCount(5)).Render()
```

SORTING.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Note: Grid columns are sorted in the **Ascending** order. If you click the already sorted column, the sort direction toggles.

 You can apply and clear sorting by invoking **sortColumn** and **clearSorting** methods.

 To disable sorting for a particular column, set the [AllowSorting](#) property of [Column](#) to false.

Initial sort

To sort at initial rendering, set the **Field** and **Direction** in [Columns](#) property of [SortSettings](#).

CSHTML

```
@{
    List<object> cols = new List<object>();
    cols.Add(new { field = "OrderDate", direction = "Ascending" });
}
```

```

        cols.Add(new { field = "Freight", direction = "Descending" });
    }
    @Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasources)
    .AllowSorting().Columns(col =>
    {
        col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
        col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

        col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
    }).AllowPaging().PageSettings(page => page.PageCount(2)).SortSettings(sort
=> sort.Columns(cols)).Render()

```

INITIAL-SORT.CS

```

public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasources = orders;
    return View();
}

```

Multi-column sorting

You can sort more than one column in a Grid. To sort multiple columns, press and hold the **CTRL** key and click the column header. The sorting order will be displayed in the header while performing multi-column sorting.

To clear sorting for a particular column, press the "Shift + mouse left click".

Note: The [AllowSorting](#) must be **true** while enabling multi-column sort.

 Set [AllowMultiSorting](#) property as **false** to disable multi-column sorting.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasources)
.AllowSorting().AllowMultiSorting().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging().PageSettings(page => page.PageCount(5)).Render()

```

MULTI-COLUMN-SORT.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Sort order

By default, the sorting order will be as **ascending -> descending -> none**.

When first click a column header it sorts the column in ascending. Again click the same column header, it will sort the column in descending. A repetitive third click on the same column header will clear the sorting.

Sort foreign key column based on Text

For local data in Grid, sorting will be performed based on the [ForeignKeyValue](#) property of [Column](#).

For remote data in Grid, sorting will be performed based on the [ForeignKeyField](#) instead of [ForeignKeyValue](#). To sort a column based on the displayed text and not based on the [ForeignKeyField](#), we need to handle the sorting operation at the server side.

The following code example describes the handling of sorting operation at the server side.

CSHTML

```
@Html.EJS().Grid("ItemGrid").DataSource(dataManger => {
    dataManger.Url("/OData/Items").Adaptor("ODataV4Adaptor"); }).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign("Right").Add();

    col.Field("EmployeeID").ForeignKeyField("EmployeeID").ForeignKeyValue("First
Name").DataSource(dm => { dm.Url("/OData/Brands").Adaptor("ODataV4Adaptor");
}).HeaderText("First Name").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn("Right").Add();
    col.Field("ShipName").HeaderText("Ship Name").Add();

}).AllowPaging().AllowSorting(true).AllowMultiSorting(true).Render()
```

FOREIGN-SORT.CS

```
public class ItemsController : ODataController
{
    [EnableQuery]
    public IQueryable<Item> Get()
    {
        List<Item> GridData =
        JsonConvert.DeserializeObject<Item[]>(Properties.Resources.ItemsJson).AsQuer
yable().ToList();
        List<Brand> empData =
        JsonConvert.DeserializeObject<Brand[]>(Properties.Resources.BrandsJson).AsQu
eryable().ToList();
    }
}
```

```

        var queryString = HttpContext.Current.Request.QueryString;
        var allUrlKeyValues =
ControllerContext.Request.GetQueryNameValuePairs();
        string key = allUrlKeyValues.LastOrDefault(x => x.Key ==
"$orderby").Value;
        if (key != null)
        {
            if (key == "EmployeeID") {
                GridData = SortFor(key); //Only for foreignKey Column
ascending
            }
            else if (key == "EmployeeID desc") {
                GridData = SortFor(key); //Only for foreignKey Column
descending
            }
        }
        var count = GridData.Count();
        var data = GridData.AsQueryable();
        return data;
    }

    public List<Item> SortFor(String Sorted)
    {
        List<Item> GridData =
JsonConvert.DeserializeObject<Item[]>(Properties.Resources.ItemsJson).AsQuer
yable().ToList();
        List<Brand> empData =
JsonConvert.DeserializeObject<Brand[]>(Properties.Resources.BrandsJson).AsQu
eryable().ToList();
        if (Sorted == "EmployeeID") //check whether ascending or descending
            empData = empData.OrderBy(e => e.FirstName).ToList();
        else if (Sorted == "EmployeeID desc")
            empData = empData.OrderByDescending(e => e.FirstName).ToList();
        List<Item> or = new List<Item>();
        for (int i = 0; i < empData.Count(); i++) {
            //Select the Field matching records
            IEnumerable<Item> list = GridData.Where(pred => pred.EmployeeID
== empData[i].EmployeeID).ToList();
            or.AddRange(list);
        }
        return or;
    }
}

```

Sorting events

During the sort action, the grid component triggers two events. The [ActionBegin](#) event triggers before the sort action starts, and the [ActionComplete](#) event triggers after the sort action is completed. Using these events you can perform the needed actions.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasources)
.AllowSorting(true).ActionComplete("sortEvent").ActionBegin("sortEvent").Col
umns(col =>
{

```

```

col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging().PageSettings(page => page.PageCount(5)).Render()
<script>
    function sortEvent(args){
        alert(args.requestType + ' ' + args.type); //custom Action
    }
</script>

```

SORT-EVENT.CS

```

public IActionResult Index()
{
    ViewBag.datasource = OrdersDetails.GetAllRecords();
    return View();
}

```

Note: The **args.requestType** is the current action name. For example, in sorting the **args.requestType** is sorting.

Touch interaction

When you tap the grid header on touchscreen devices, the selected column header is sorted. A popup

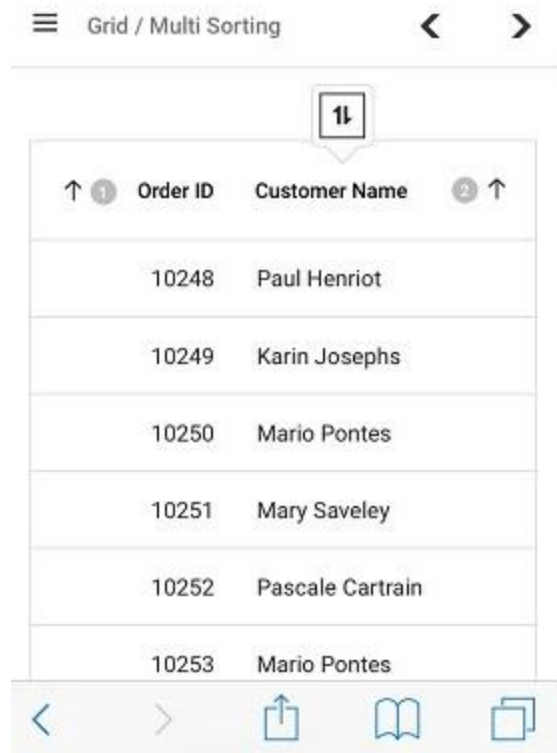


is displayed for multi-column sorting. To sort multiple columns, tap the popup , and then tap the desired grid headers.



Note: The [AllowMultiSorting](#) and [AllowSorting](#) should be **true** then only the popup will be shown.

The following screenshot shows grid touch sorting.



See Also

- [How to perform own sorting logic in ASP.NET MVC Grid](#)

Grouping in ASP.Net MVC Grid Component

The Grid has options to group records by dragging and dropping the column header to the group drop area. When grouping is applied, grid records are organized into a hierarchical structure to facilitate easier expansion and collapse of records.

To enable grouping in the grid, set the [AllowGrouping](#) as true. Grouping options can be configured through the [GroupSettings](#).

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowGrouping().Render()
```


GROUP.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: You can group and ungroup columns by using the [groupColumn](#) and [ungroupColumn](#) methods.

 To disable grouping for a particular column, set [AllowGrouping](#) property of [Column](#) to false.

Initial group

To apply group at initial rendering, set the column field name in the [Columns](#) property of [GroupSettings](#).

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowGrouping().GroupSettings(group => { group.Columns(new string[] {
"CustomerID" }); }).Render()
```

INITIAL-GROUP.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Hide drop area

To avoid ungrouping or further grouping of a column after initial column grouping, define the [ShowDropArea](#) of [GroupSettings](#) as false.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
}
```

```
col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowGrouping().GroupSettings(group => {
group.ShowDropArea(false).Columns(new string[] { "Freight" }); }).Render()
```

HIDE-DROP-AREA.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Group with paging

On grouping columns with paging feature, the aggregated information and total items are displayed based on the current page. The grid does not consider aggregated information and total items from other pages. To get additional details (aggregated information and total items) from other pages, set the [DisablePageWiseAggregates](#) of [GroupSettings](#) to false.

Note: If remote data is bound to grid dataSource, two requests will be sent when performing grouping action; one for getting the grouped data and another for getting aggregate details and total items count.

Group by format

By default, a column will be grouped by the data or value present for the particular row. To group the numeric or datetime column based on the mentioned format, you have to enable the [EnableGroupByFormat](#) property of the corresponding grid columns.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").EnableGroupByFormat(true).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).EnableGroupByFormat(true).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowGrouping().GroupSettings(group => {
group.ShowDropArea(false).Columns(new string[] { "OrderDate", "Freight" });
}).Render()
```

GROUP-FORMAT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Grouping events

During the group action, the grid component triggers two events. The [ActionBegin](#) event triggers before the group action starts and the [ActionComplete](#) event triggers after the group action is completed. Using these events you can perform any action.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.dataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
})
.AllowPaging().AllowGrouping().ActionBegin("actionHandler").ActionComplete
("actionHandler").Render()
<script>
    function actionHandler(args) {
        alert(args.requestType + ' ' + args.type); //Custom Action
    }
</script>
```

GROUPING-EVENTS.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: The **args.requestType** is based on the current action name. For example, when grouping, the **args.requestType** value will be grouping.

Collapse by external button

You can collapse the selected group from an external button by invoking the `expandCollapseRows` method.

CSHTML

```
@Html.EJS().Button("collapse").Content("Collapse").IsPrimary(true).Render()
```

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowGrouping().GroupSettings(group => { group.Columns(new string[] {
"Freight" }).ShowDropArea(false); }).Render()
<script>
    document.getElementById('collapse').addEventListener('click', () => {
        var grid = document.getElementById('Grid').ej2_instances[0];
        if (grid.getSelectedRowIndex().length) {
            var currentTr = grid.getRows()[grid.getSelectedRowIndex()[0]];
            while (currentTr.classList && currentTr.classList.length) {
                currentTr = currentTr.previousSibling;
            }
            let collapseElement = currentTr.querySelector('.e-
recordplusexpand');
            grid.groupModule.expandCollapseRows(collapseElement); //Pass the
collapse row element.
        }
    });
</script>
```

COLLAPSE.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Sort grouped columns in descending order during initial grouping

By default, grouped columns are sorted in ascending order. To sort grouped columns in descending order during initial grouping, you can set the **field** and **direction** in the [Columns](#) property of **SortSettings**.

The **CustomerID** column will be sorted in descending order when the grid is initially grouped, as shown in the following example.

CSHTML

```
@{
    List<object> cols = new List<object>();
    cols.Add(new { field = "CustomerID", direction = "Descending" });
}
```

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowGrouping().AllowSorting().SortSettings(sort =>
sort.Columns(cols)).GroupSettings(group => { group.Columns(new string[] {
"CustomerID" }); }).Render()
```

SORT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

See Also

- [Exporting grouped records](#)

Filtering in ASP.Net MVC Grid Component

Filtering allows you to view particular records based on filter criteria. To enable filtering in the Grid, set the [AllowFiltering](#) to true. Filtering options can be configured through [FilterSettings](#) property.

<!--

Grid supports two types of filter, they are:

- Filter bar
- Excel

-->

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowFiltering(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
```

```
col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Render()
```

FILTER.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: You can apply and clear filtering by using [filterByColumn](#) and [clearFiltering](#) methods.

 To disable filtering for a particular column, set [AllowFiltering](#) of [Column](#) to false.

Initial filter

To apply the filter at initial rendering, set the filter **Predicate** object in [FilterSettings](#) property of [Column](#).

CSHTML

```
@{
    List<object> filterColumns = new List<object>();
    filterColumns.Add(new { field = "EmployeeID", matchCase = false,
@operator = "equal", predicate = "and", value = 1 });
    filterColumns.Add(new { field = "ShipCountry", matchCase = false,
@operator = "startswith", predicate = "and", value = "Denmark" });
}
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowFiltering(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("EmployeeID").HeaderText("Employee ID").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).FilterSettings(filter =>
filter.Columns(filterColumns)).AllowPaging().Render()
```

INITIALFILTER.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Filter operators

The filter operator for a column can be defined in the [Operators](#) property of [FilterSettings](#).

The available operators and its supported data types are:

Operator	Description	Supported Types
----- ----- -----		
startswith	Checks whether the value begins with the specified value.	String
endswith	Checks whether the value ends with the specified value.	String
contains	Checks whether the value contains the specified value.	String
doesnotstartwith	Checks whether the value does not begin with the specified value.	String
doesnotendwith	Checks whether the value does not end with the specified value.	String
doesnotcontain	Checks whether the value does not contain the specified value.	String
equal	Checks whether the value is equal to the specified value.	String | Number | Boolean | Date
notequal	Checks for values not equal to the specified value.	String | Number | Boolean | Date
greaterthan	Checks whether the value is greater than the specified value.	Number | Date
greaterthanorequal	Checks whether a value is greater than or equal to the specified value.	Number | Date
lessthan	Checks whether the value is less than the specified value.	Number | Date
lessthanorequal	Checks whether the value is less than or equal to the specified value.	Number | Date
isnull	Returns the values that are null.	String | Number | Date
isnotnull	Returns the values that are not null.	String | Number | Date
isempty	Returns the values that are empty.	String
isnotempty	Returns the values that are not empty.	String
between	Filter the values based on the range between the start and end specified values.	Number | Date

Wildcard and LIKE operator filter

Wildcard and **LIKE** filter operators filters the value based on the given string pattern, and they apply to string-type columns. But it will work slightly differently.

Wildcard filtering

The **Wildcard** filter can process one or more search patterns using the "*" symbol, retrieving values matching the specified patterns.

- The **Wildcard** filter option is supported for the DataGrid that has all search options.

For example:

Operator | Description

%ab% | Returns all the value that are contains "ab" character.

ab% | Returns all the value that are ends with "ab" character.

%ab | Returns all the value that are starts with "ab" character.

![LIKEFilter](../images/like_filter.gif)

Note: By default, the [Operators](#) value is **equal**.

See Also

- [Customizing Filter Dialog by using an additional parameter](#)
- [Hide sorting options on Excel filter Dialog](#)

Search in ASP.NET MVC Grid Component

You can search records in a Grid, by using the [search](#) method with search key as a parameter. This also provides an option to integrate search text box in grid's toolbar by adding **Search** item to the [Toolbar](#).

Note: The clear icon is shown in the Data Grid search text box when it is focused on search text or after typing the single character in the search text box. A single click of the clear icon clears the text in the search box as well as the search results in the Data Grid.

CSHTML

```
@Html.EJS().Grid("Search").DataSource((IEnumerable<object>)ViewBag.datasourc
e).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("160").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("170").Add();

    col.Field("Freight").HeaderText("Freight").Width("170").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Format("C2").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("170").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
}).AllowPaging().PageSettings(page => page.PageCount(2)).Toolbar(new
List<string>() { "Search" }).Render()
```

SEARCH.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Initial search

To apply search at initial rendering, set the fields, operator, key, and ignoreCase in the [SearchSettings](#) property.

CSHTML

```
@Html.EJS().Grid("Search").DataSource((IEnumerable<object>)ViewBag.datasourc
e).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("160").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("170").Add();

    col.Field("Freight").HeaderText("Freight").Width("170").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Format("C2").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("170").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
}).AllowPaging().SearchSettings(search => { search.Fields(new string[] {
"CustomerID" }).Key("Ha").Operator("contains").IgnoreCase(true);
}).PageSettings(page => page.PageCount(2)).Toolbar(new List<string>() {
"Search" }).Render()
```

INITIAL-SEARCH.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Note: By default, grid searches all the bound column values. To customize this behavior define the [Fields](#) of [SearchSettings](#) property.

Search operators

The search operator can be defined in the [Operator](#) property of [SearchSettings](#) to configure specific searching.

The following operators are supported in searching:

Operator	Description
----- -----	
startswith	Checks whether a value begins with the specified value.
endswith	Checks whether a value ends with the specified value.
contains	Checks whether a value contains the specified value.
equal	Checks whether a value is equal to the specified value.
notequal	Checks for values not equal to the specified value.

Note: By default, the [Operator](#) value is **contains**.

Search by external button

To search grid records from an external button, invoke the [search](#) method.

CSHTML

```
@Html.EJS().Button("search").Content("Search").Render()
@Html.EJS().Grid("Search").DataSource((IEnumerable<object>)ViewBag.datasourc
e).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("160").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("170").Add();

    col.Field("Freight").HeaderText("Freight").Width("170").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Format("C2").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("170").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
}).AllowPaging().PageSettings(page=>page.PageCount(2)).Render()
<script>
    document.getElementById('search').addEventListener('click', () => {
        var gridObj = document.getElementById("Search").ej2_instances[0];
        var searchString = "AL";
        gridObj.search(searchString);
    });
</script>
```

EXTERNAL-BTN.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Search specific columns

By default, grid searches all visible columns. You can search specific columns by defining the specific column's field names in the [Fields](#) property of [SearchSettings](#).

CSHTML

```
@Html.EJS().Grid("Search").DataSource((IEnumerable<object>)ViewBag.datasourc
e).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("160").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("170").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("170").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("140").Add();
}).AllowPaging().SearchSettings(search => { search.Fields(new string[] {
"CustomerID", "ShipCity", "ShipName" }); }).PageSettings(page =>
page.PageCount(2)).Toolbar(new List<string>() { "Search" }).Render()
```

SEARCH-A-COLUMN.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Clear search by external button

To clear the searched grid records from the external button, set [Key](#) property as **empty** string.

CSHTML

```
@Html.EJS().Button("clear").Content("Clear Search").Render()
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasources)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("160").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("170").Add();

    col.Field("Freight").HeaderText("Freight").Width("170").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Format("C2").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("170").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
}).AllowPaging().SearchSettings(search => { search.Fields(new string[] {
"CustomerID" }).Key("Ha").Operator("contains").IgnoreCase(true);
}).PageSettings(page => page.PageCount(2)).Toolbar(new List<string>() {
"Search" }).Render()
<script>
    document.getElementById('clear').addEventListener('click', () => {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        gridObj.searchSettings.key = '';
    });
</script>
```

CLEAR-SEARCH.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasources = orders;
    return View();
}
```

Search on each key stroke

You can search the Grid data on each key stroke by binding the **keyup** event for the search input element inside the [created](#) event. Inside the **keyup** handler you can search the Grid by invoking the [search](#) method of the Grid component.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasources)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("160").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("170").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("170").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("140").Add();
}).AllowPaging().Created("created").Toolbar(new List<string>() { "Search"
}).Render()
<script>
```

```
function created() {
    var grid = document.getElementById("Grid").ej2_instances[0];
    document.getElementById(grid.element.id +
        "_searchbar").addEventListener('keyup', () => {
        grid.search(event.target.value)
    });
}
</script>
```

SEARCH-EACH-KEY.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Perform search operation in Grid using multiple keywords

You can perform a searching operation in the Grid using multiple keywords. This can be achieved by the [ActionBegin](#) event of the Grid.

In the following sample, we have performed the searching with multiple keywords by using the query property of grid when the requestType is searching in the [ActionBegin](#) event.

CSHTML

```
@using dotnet_mvc_sample.Controllers;
@{
    List<object> toolbarItems = new List<object>();
    toolbarItems.Add("Search");
}
@(Html.EJS().Grid<HomeController.BigData>("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("OrderID").IsPrimaryKey(true).Width("120").Add();
    col.Field("CustomerID").HeaderText("CustomerID").Width("120").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").Add();
    col.Field("ShipCity").HeaderText("ShipCity").Width("120").Add();
}).Toolbar(toolbarItems).ActionBegin("actionBegin").ActionComplete("actionComplete").Render())
<script>
    var values;
    var key = '';
    var refresh = false;
    var removeQuery = false;
    var valueAssign = false;
    function actionBegin(args) {
        if (args.requestType == 'searching') {
            const keys = args.searchString.split(',');
            var flag = true;
            var predicate;
```

```

        if (keys.length > 1) {
            if (this.searchSettings.key !== '') {
                values = args.searchString;
                keys.forEach((key) => {
                    this.getColumns().forEach((col) => {
                        if (flag) {
                            predicate = new ej.data.Predicate(
                                col.field,
                                'contains',
                                key,
                                true
                            );
                            flag = false;
                        } else {
                            var predic = new ej.data.Predicate(
                                col.field,
                                'contains',
                                key,
                                true
                            );
                            predicate = predicate.or(predic);
                        }
                    });
                });
            }
            this.query = new ej.data.Query().where(predicate);
            this.searchSettings.key = '';
            refresh = true;
            valueAssign = true;
            removeQuery = true;
            this.refresh();
        }
    }
}

function actionComplete(args) {
    if (args.requestType === 'refresh' && valueAssign) {
        document.getElementById(this.element.id + '_searchbar').value =
values;
        valueAssign = false;
    } else if (
document.getElementById(this.element.id + '_searchbar').value
=== '' &&
args.requestType === 'refresh' &&
removeQuery
) {
        this.query = new ej.data.Query();
        removeQuery = false;
        this.refresh();
    }
}
</script>

```

SEARCH-MULTIPLE-KEYWORDS.CS

```

public IActionResult Index()
{

```

```
var orders = OrderDetails.GetAllRecords();
ViewBag.Datasource = orders;
return View();
}
```

Note: * Define multiple keywords by using a comma separator in search bar to search.

Note: * Search operation can be performed in foreign key column based on following way.

 * When a value is searched on a grid with the foreign key column, a filter query is sent to the foreign key data source, and the appropriate column is filtered depending on the search value. The search query will be sent to the grid data source, and the value of the associated field will be returned.

See Also

- [Add clear icon in search in ASP.NET MVC Grid](#)
- [How to perform search by using Wildcard and LIKE operator filter](#)

Paging

Paging provides an option to display Grid data in page segments. To enable paging, set the [AllowPaging](#) to true. When paging is enabled, pager component renders at the bottom of the grid. Paging options can be configured through the [PageSettings](#).

In the below sample, **pageSize** is calculated based on the grid height by using the [Load](#) event.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Height("325").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Load("load").Render()
<script>
    function load() {
        var grid = document.getElementById('Grid').ej2_instances[0];
        var rowHeight = grid.getRowHeight(); //height of the each row
        var gridHeight = grid.height; //grid height
        var pageSize = grid.pageSettings.pageSize; //initial page size
        var pageResize = (gridHeight - (pageSize * rowHeight)) / rowHeight;
        //new page size is obtained here
        grid.pageSettings.pageSize = pageSize + Math.round(pageResize);
    }
</script>
```

PAGE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Note: You can achieve better performance by using grid paging to fetch only a pre-defined number of records from the data source.

Template

You can use custom elements inside the pager instead of default elements. The custom elements can be defined by using the [Template](#) property of [PageSettings](#). Inside this template, you can access the **CurrentPage**, **PageSize**, **TotalPage** and **TotalRecordCount** values.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().PageSettings(page => {
    page.PageSize(7).Template("#template");
}).ActionComplete("actionComplete").DataBound("dataBind").Render()
<script>
    function updateTemplate() {
        var numeric;
        var grid = document.getElementById("Grid").ej2_instances[0];
        numeric = new ej.inputs.NumericTextBox({
            min: 1,
            max: 3,
            step: 1,
            width: 100,
            format: '###.##',
            change: function (args) {
                let value = args.value;
                grid.goToPage(value);
            }
        });
        numeric.appendTo('#currentPage');
    };
    var flag = true;
    function dataBind() {
        if (flag) {
            flag = false;
            updateTemplate();
        }
    }
}
```

```

function actionComplete(args) {
    if (args.requestType === 'paging') {
        updateTemplate();
    }
}
</script>
<script id="template" type="text/x-template">
    <div class="e-pagertemplate">
        <div>
            <div class="content-wrapper">
                <input id="currentPage" type="text" value=${currentPage}>
            </div>
        </div>
        <div id="totalPages" class="e-pagertemplatemessage" style="margin-top:5px;margin-left:30px;border: none; display: inline-block ">
            <span class="e-pagenomsg">${currentPage} of ${totalPages} pages
            (${totalRecordsCount} items)</span>
        </div>
    </div>
</script>

```

PAGE-TEMP.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}

```

Pager with Page Size Dropdown

The pager Dropdown allows you to change the number of records in the Grid dynamically. It can be enabled by defining the [PageSizes](#) property of [PageSettings](#) as true.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().PageSettings(page=> { page.PageSizes(true); }).Render()

```

PAGERDROPDOWN.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
}

```



```
return View();
}
```

How to render Pager at the Top of the Grid

By default, Pager will be rendered at the bottom of the Grid. You can also render the Pager at the top of the Grid by using the [DataBound](#) event.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
})
.AllowPaging().DataBound("dataBound").Toolbar(new List<string>() { "Add",
"Edit", "Delete", "Update", "Cancel" }).Render()
<script>
    var initialGridLoad = true;
    function dataBound() {
        if (initialGridLoad) {
            initialGridLoad = false;
            var pager = document.getElementsByClassName('e-gridpager');
            var toolbar = document.getElementsByClassName('e-toolbar');
            this.element.insertBefore(pager[0], toolbar[0]);
        }
    }
</script>
```

PAGERATTOP.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Note: During the paging action, the pager component triggers the below three events.

 The created event triggers when Pager is created.

 The click event triggers when the numeric items in the pager is clicked.

 The dropDownChanged event triggers when pageSize DropDownList value is selected.

See Also

- [Group with Paging](#)

Scrolling

The scrollbar will be displayed in the grid when content exceeds the element [Width](#) or [Height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the grid exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the grid element width.
- The [Height](#) and [Width](#) are used to set the grid height and width, respectively.

Note: The default value for [Height](#) and [Width](#) is **auto**.

Set width and height

To specify the [Width](#) and [Height](#) of the scroller in the pixel, set the pixel value to a number.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Height("400").Width("auto").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipAddress").HeaderText("Ship Address").Width("170").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("170").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).Render()
```

WIDTH-HEIGHT.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Responsive with parent container

Specify the [Width](#) and [Height](#) as **100%** to make the grid element fill its parent container. Setting the [Height](#) to **100%** requires the grid parent element to have explicit height.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Height("100%").Width("auto").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipAddress").HeaderText("Ship Address").Width("170").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("170").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).Render()
```

RESPONSIVE-PARENT.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Scroll to selected row

You can scroll the grid content to the selected row position by using the [RowSelected](#) event.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Placeholder("Enter index to select a
row").Change("onChange").Width("200").Render()
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Height("400").Width("auto").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipAddress").HeaderText("Ship Address").Width("170").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("170").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}
```

```

}).RowSelected("rowSelected").Render()
<script>
    function onChange(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        gridObj.selectRow(parseInt(this.getText(), 10));
    }
    function rowSelected(args) {
        var rowHeight =
this.getRows()[this.getSelectedRowIndex()[0]].scrollHeight;
        this.getContent().children[0].scrollTop = rowHeight *
this.getSelectedRowIndex()[0];
    }
</script>

```

SCROLL-SELECTED-ROW.CS

```

public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}

```

Hide the scrollbar when the content is not overflown

You can hide the scrollbar of Grid content by using the [hideScroll](#) method when the content doesn't overflow its parent element.

In the following sample, we have invoked the [hideScroll](#) method inside the [dataBound](#) event.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Height("400").Width("auto").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
}).DataBound("dataBound").AllowPaging().PageSettings(page=> {
page.PageSize(2); }).Render()
<script>
    function dataBound() {
        var grid = document.getElementById('Grid').ej2_instances[0];
        grid.hideScroll();
    }
</script>

```

HIDE-SCROLL.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Frozen rows and columns

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the grid while scrolling.

In this demo, the [FrozenColumns](#) is set as **2** and the [FrozenRows](#) is set as **3**. Hence, the left two columns and top three rows are frozen.

CSHTML

```
@Html.EJS().Grid("FrozenGrid").DataSource((IEnumerable<object>)ViewBag.datasource).AllowSelection(false).AllowResizing().Height("410").FrozenRows(2).FrozenColumns(1).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("200").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShippedDate").HeaderText("Shipped Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("200").Add();
    col.Field("ShipAddress").HeaderText("Ship Address").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).Render()
```

FREEZE-ROW-COLUMN.CS

```
public IActionResult Index()
{
    ViewBag.datasource = OrdersDetails.GetAllRecords();
    return View();
}
```

Note: Frozen rows and columns should not be set outside the grid view port.

 Frozen Grid will support row and column virtualization feature, which helps to improve the Grid performance while loading a large dataset.

Limitations of Frozen Grid

The following features are not supported in frozen rows and columns:

- Detail Template
- Hierarchy Grid

Freeze particular columns

To freeze particular column in the grid, the [IsFrozen](#) property can be used.

In this demo, the columns with field name **OrderID** and **CustomerID** is frozen using the [IsFrozen](#) property.

CSHTML

```
@Html.EJS().Grid("FrozenGrid").DataSource((IEnumerable<object>)ViewBag.datas
ource).AllowSelection(false).AllowResizing().Height("410").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").IsFrozen(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Ri
ght).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").IsFrozen(true).Width("200").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("200").Add();
    col.Field("ShipAddress").HeaderText("Ship Address").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).Render()
```

FREEZE-COLUMN.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Freeze Direction

You can freeze the Grid columns on the left or right side by using the [column.freeze](#) property and the remaining columns will be movable. The grid will automatically move the columns to the left or right position based on the [column.freeze](#) value.

Types of the [column.freeze](#) directions:

- **Left:** Allows you to freeze the columns at the left.
- **Right:** Allows you to freeze the columns at the right.

- **Fixed:** Allows you to lock the column at a fixed position by ensuring its visibility during horizontal scroll.

In this demo, the **ShipCountry** column is frozen at the left and the **CustomerID** column is frozen at the right side of the content table.

CSHTML

```
@Html.EJS().Grid("FrozenGrid").DataSource((IEnumerable<object>)ViewBag.datas
ource).FrozenRows(2).Height("410").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
ID").Width("150").Freeze(Syncfusion.EJ2.Grids.FreezeDirection.Right).Add();
    col.Field("EmployeeID").HeaderText("Employee
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipAddress").HeaderText("Ship Address").Width("150").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").Freeze(Syncfusion.EJ2.Grids.FreezeDirection.Left).Add
();
}).Render()
```

FREEZE-COLUMN.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Note: * Freeze Direction is not compatible with the `isFrozen` and `frozenColumns` properties.

Deprecated Methods

Deprecated Methods | [Previous](#) | [Current](#) | [Suggested Alternative Methods](#) | [Example for achieving the same results](#)

getMovableRows() | In the previous architecture of frozen grid, three separate tables were created for the left, right, and movable contents. When calling this method, it would return only the movable table rows (tr's). | In the current architecture, the frozen left, right, and movable sections are applied within a single table. When calling this method, it will return all table rows (tr's) of the entire table. However, in this approach, we have introduced the `e-unfreeze` class for movable cells. This allows us to selectively retrieve the movable rows using the `e-unfreeze` class selector. | **getRows()** |

`gridInstance.getMovableRows()[0].querySelectorAll('.e-unfreeze')` // Deprecated
 (or)
`gridInstance.getRows()[0].querySelectorAll('.e-unfreeze')` // Alternative method

`getFrozenRightRows()` | In the previous architecture, this method would return only the table rows (tr's) from the freeze right table. | In the current architecture, the frozen left, right, and movable sections are applied within a single table. When calling this method, it will return all the rows (tr's) of the entire table. In this new approach, we have introduced the `e-rightfreeze` class for right freeze cells. As a result, you can now selectively retrieve the right freeze rows using the `e-rightfreeze` class selector.
 | `getRows()` | `gridInstance.getFrozenRightRows()[0].querySelectorAll('.e-rightfreeze')` // Deprecated
 (or)
`gridInstance.getRows()[0].querySelectorAll('.e-rightfreeze')` // Alternative method

`getMovableRowByIndex()` (or) `getFrozenRowByIndex()` (or) `getFrozenRightRowByIndex()` | In the previous architecture, you could select rows by using separate methods for each table section. Like,
 (or) `getMovableRowByIndex` - select a movable row (or) `getFrozenRowByIndex` - select a freeze row (or) `getFrozenRightRowByIndex` - select a right freeze row. | In the current architecture, the `getMovableRowByIndex`, `getFrozenRightRowByIndex` and `getFrozenRowByIndex` methods all return the same table row (tr) based on the given index. Additionally, class names for table cells (td's) have been separated as follows: (or) Left-Freeze : `e-leftfreeze` (or) Movable : `e-unfreeze` (or) Right-Freeze : `e-rightfreeze`. This separation of class names makes it easier to target and customize the cells within the particular row. | `getRowByIndex()` | **To get the left freeze cells:**
`gridInstance.getRowByIndex(1).querySelectorAll('.e-leftfreeze')` (or) **To get the movable cells:**
`gridInstance.getRowByIndex(1).querySelectorAll('.e-unfreeze')` (or) **To get the right freeze cells:**
`gridInstance.getRowByIndex(1).querySelectorAll('.e-rightfreeze')`

`getMovableCellFromIndex()` (or) `getFrozenRightCellFromIndex()` | `getMovableCellFromIndex()` - select a particular cell in the movable table. (or) `getFrozenRightCellFromIndex()` - select a particular cell in the right freeze table. | In the new approach, you can select a particular cell by using both the `getFrozenRightCellFromIndex` and `getMovableCellFromIndex` methods. | `getCellFromIndex()`
`gridInstance.getCellFromIndex(1,1)`

`getMovableDataRows()` (or) `getFrozenRightDataRows()` (or) `getFrozenDataRows()` | These methods returns the viewport data rows for the freeze, movable, and right tables separately. | In the new approach, when calling the `getMovableDataRows`, `getFrozenRightDataRows`, and `getFrozenDataRows` methods, returns the entire viewport data rows. You can then select specific cells within these rows using the following selectors (or) Left-Freeze : `e-leftfreeze` (or) Movable : `e-unfreeze` (or) * Right-Freeze : `e-rightfreeze`. | `getDataRows()` | **To get the movable data cells:**
`gridInstance.getDataRows()[0].querySelectorAll('.e-unfreeze')` (or) **To get the right freeze data cells:**
`gridInstance.getDataRows()[0].querySelectorAll('.e-rightfreeze')` (or) **To get the left freeze data cells:**
`gridInstance.getDataRows()[0].querySelectorAll('.e-leftfreeze')`

`getMovableColumnHeaderByIndex()` (or) `getFrozenRightColumnHeaderByIndex()` (or) `getFrozenLeftColumnHeaderByIndex()` | In the previous architecture, these methods selects the movable, right freeze, and left freeze headers from the table separately. | In the new approach, when calling the `getMovableColumnHeaderByIndex`, `getFrozenRightColumnHeaderByIndex`, and `getFrozenLeftColumnHeaderByIndex` methods, you will still receive the same results as before. | `getColumnHeaderByIndex()` | `gridInstance.getColumnHeaderByIndex(1)`

When a validation message is displayed in the frozen part (Left, Right, Fixed) of the table, scrolling is prevented until the validation message is cleared.

Virtualization in ASP.NET MVC Grid Component

Grid allows you to load large amount of data without performance degradation.

Row Virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the data will be loaded while scrolling vertically. To setup the row virtualization, you need to define [EnableVirtualization](#) as true and content height by [Height](#) property.

The number of records displayed in the Grid is determined implicitly by height of the content area. Also, you have an option to define a visible number of records by the [PageSettings.PageSize](#) property. The loaded data will be cached and reused when it is needed for next time.

CSHTML

```
@Html.EJS().Grid("VirtualGrid").DataBound("hide").EnableVirtualization().Height("500").Columns(col =>
{
    col.Field("Field1").HeaderText("S.No").IsPrimaryKey(true).ValidationRules(new { required = "true" }).Width("140").Add();

    col.Field("Field2").HeaderText("Year").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field3").HeaderText("Stint").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field4").HeaderText("TMID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field5").HeaderText("LGID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).EditSettings(edit => {
    edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Normal);
}).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel" }).Render()

<script>
    var virtualData = [], date1=null, date2=null, flag= true;
    var names = ['hardire', 'abramjo01', 'aubucch01', 'Hook', 'Rumpelstiltskin', 'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa', 'Anna', 'Snow White', 'Prince Charming', 'Cora', 'Zelena', 'August', 'Mulan', 'Graham', 'Discord', 'Will', 'Robin Hood', 'Jiminy Cricket', 'Henry', 'Neal', 'Red', 'Aaran', 'Aaren', 'Aarez', 'Aarman', 'Aaron', 'Aaron-James', 'Aarron', 'Aaryan', 'Aaryn', 'Aayan', 'Aazaan', 'Abaan', 'Abbas', 'Abdallah', 'Abdalroof', 'Abdihakim', 'Abdirahman', 'Abdisalam', 'Abdul', 'Abdul-Aziz', 'Abdulbasir', 'Abdulkadir', 'Abdulkarem', 'Abdulkhader', 'Abdullah', 'Abdul-Majeed', 'Abdulmalik', 'Abdul-Rehman', 'Abdur', 'Abdurraheem', 'Abdur-Rahman', 'Abdur-Rehmaan', 'Abel', 'Abhinav', 'Abhisumant', 'Abid', 'Abir', 'Abraham', 'Abu', 'Abubakar', 'Ace', 'Adain', 'Adam', 'Adam-James', 'Addison', 'Addisson', 'Adegbola', 'Adegbolahan', 'Aden', 'Adenn', 'Adie', 'Adil', 'Aditya',
```

```

        'Adnan', 'Adrian', 'Adrien', 'Aedan', 'Aedin', 'Aedyn', 'Aeron',
        'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-Aziz', 'Ahoua', 'Ahtasham',
        'Aiadan', 'Aidan', 'Aiden', 'Aiden-Jack', 'Aiden-Vee'];
        document.getElementById("generate").addEventListener('click', () =>
    {
        var grid =
document.getElementById("VirtualGrid").ej2_instances[0]
        if (!virtualData.length) {
            show();
            dataSource();
            date1 = new Date().getTime();
            grid.dataSource = virtualData;
        } else {
            flag = true;
            show();
            date1 = new Date().getTime();
            grid.refresh();
        }
    })
    function show() {
        document.getElementById('popup').style.display = 'inline-block';
    }
    function hide(args) {
        if (flag && date1) {
            var date2 = new Date().getTime();
            document.getElementById('performanceTime').innerHTML = 'Time
Taken: ' + (date2 - date1) + 'ms';
            flag = false;
        }
        document.getElementById('popup').style.display = 'none';
    }
    function dataSource() {
        for (var i= 0; i < 100000; i++) {
            virtualData.push({
                'Field1': i + 1,
                'Field2': 1967 + (i % 10),
                'Field3': Math.floor(Math.random() * 200),
                'Field4': Math.floor(Math.random() * 100),
                'Field5': Math.floor(Math.random() * 2000),
            });
        }
    }
}
</script>

```

ROW-VIRTUALIZATION.CS

```

public IActionResult Index()
{
    return View();
}

```

Column Virtualization

Column virtualization allows you to virtualize columns. It will render columns which are in the viewport. You can scroll horizontally to view more columns.

To setup the column virtualization, set the [EnableVirtualization](#) and [EnableColumnVirtualization](#) properties as **true**.

CSHTML

```
@Html.EJS().Grid("VirtualGrid").DataBound("hide").EnableColumnVirtualization
().EnableVirtualization().Height("500").Columns(col =>
{
    col.Field("Field1").HeaderText("S.No").Width("140").IsPrimaryKey(true).Valid
ationRules(new { required = "true"}).Add();

    col.Field("Field2").HeaderText("Year").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field3").HeaderText("Stint").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field4").HeaderText("TMID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field5").HeaderText("LGID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field6").HeaderText("GP").Width("120").TextAlign(Syncfusion.EJ2.G
rids.TextAlign.Right).Add();

    col.Field("Field7").HeaderText("GS").Width("120").TextAlign(Syncfusion.EJ2.G
rids.TextAlign.Right).Add();

    col.Field("Field8").HeaderText("Minutes").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field9").HeaderText("Points").Width("120").TextAlign(Syncfusion.E
J2.Grids.TextAlign.Right).Add();

    col.Field("Field10").HeaderText("oRebounds").Width("130").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field11").HeaderText("dRebounds").Width("130").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field12").HeaderText("Rebounds").Width("130").TextAlign(Syncfusio
n.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field13").HeaderText("Assists").Width("120").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field14").HeaderText("Steals").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field15").HeaderText("Blocks").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

    col.Field("Field16").HeaderText("Turnovers").Width("130").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();
```

```

col.Field("Field17").HeaderText("PF").Width("120").TextAlign(Syncfusion.EJ2.
Grids.TextAlign.Right).Add();

col.Field("Field18").HeaderText("fgAttempted").Width("150").TextAlign(Syncfu
sion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field19").HeaderText("fgMade").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

col.Field("Field20").HeaderText("ftAttempted").Width("150").TextAlign(Syncfu
sion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field21").HeaderText("ftMade").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

col.Field("Field22").HeaderText("ThreeAttempted").Width("150").TextAlign(Syn
cfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field23").HeaderText("ThreeMade").Width("130").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field24").HeaderText("PostGP").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

col.Field("Field25").HeaderText("PostGS").Width("120").TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

col.Field("Field26").HeaderText("PostMinutes").Width("120").TextAlign(Syncfu
sion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field27").HeaderText("PostPoints").Width("130").TextAlign(Syncfus
ion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field28").HeaderText("PostoRebounds").Width("130").TextAlign(Sync
fusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field29").HeaderText("PostdRebounds").Width("130").TextAlign(Sync
fusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Field30").HeaderText("PostRebounds").Width("130").TextAlign(Syncf
usion.EJ2.Grids.TextAlign.Right).Add();
}).EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel" }).Render()
<script>
    var virtualData = [], datel=null, date2=null, flag= true;
    var names = ['hardire', 'abramjo01', 'aubucch01', 'Hook',
'Rumpelstiltskin', 'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa',
'Anna', 'Snow White', 'Prince Charming', 'Cora', 'Zelena',
'August', 'Mulan', 'Graham', 'Discord', 'Will', 'Robin Hood',
'Jiminy Cricket', 'Henry', 'Neal', 'Red', 'Aaran', 'Aaren',
'Aarez', 'Aarman', 'Aaron', 'Aaron-James', 'Aarron', 'Aaryan', 'Aaryn',
'Aayan', 'Aazaan', 'Abaan', 'Abbas', 'Abdallah', 'Abdalroof',
'Abdihakim', 'Abdirahman', 'Abdisalam', 'Abdul', 'Abdul-Aziz',

```

```

        'Abdulbasir', 'Abdulkadir', 'Abdulkarem', 'Abdulkhader',
        'Abdullah', 'Abdul-Majeed', 'Abdulmalik', 'Abdul-Rehman', 'Abdur',
        'Abdurraheem', 'Abdur-Rahman', 'Abdur-Rehmaan', 'Abel',
        'Abhinav', 'Abhisumant', 'Abid', 'Abir', 'Abraham', 'Abu', 'Abubakar',
        'Ace', 'Adain', 'Adam', 'Adam-James', 'Addison', 'Addisson',
        'Adegbola', 'Adegbolahan', 'Aden', 'Adenn', 'Adie', 'Adil', 'Aditya',
        'Adnan', 'Adrian', 'Adrien', 'Aedan', 'Aedin', 'Aedyn', 'Aeron',
        'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-Aziz', 'Ahoua', 'Ahtasham',
        'Aiadan', 'Aidan', 'Aiden', 'Aiden-Jack', 'Aiden-Vee'];
        document.getElementById("generate").addEventListener('click', () =>
    {
        var grid =
document.getElementById("VirtualGrid").ej2_instances[0]
        if (!virtualData.length) {
            show();
            dataSource();
            date1 = new Date().getTime();
            grid.dataSource = virtualData;
        } else {
            flag = true;
            show();
            date1 = new Date().getTime();
            grid.refresh();
        }
    })
    function show() {
        document.getElementById('popup').style.display = 'inline-block';
    }
    function hide(args) {
        if (flag && date1) {
            var date2 = new Date().getTime();
            document.getElementById('performanceTime').innerHTML = 'Time
Taken: ' + (date2 - date1) + 'ms';
            flag = false;
        }
        document.getElementById('popup').style.display = 'none';
    }
    function dataSource() {
        for (var i= 0; i < 100000; i++) {
            virtualData.push({
                'Field1': i + 1,
                'Field2': 1967 + (i % 10),
                'Field3': Math.floor(Math.random() * 200),
                'Field4': Math.floor(Math.random() * 100),
                'Field5': Math.floor(Math.random() * 2000),
                'Field6': Math.floor(Math.random() * 1000),
                'Field7': Math.floor(Math.random() * 100),
                'Field8': Math.floor(Math.random() * 10),
                'Field9': Math.floor(Math.random() * 10),
                'Field10': Math.floor(Math.random() * 100),
                'Field11': Math.floor(Math.random() * 100),
                'Field12': Math.floor(Math.random() * 1000),
                'Field13': Math.floor(Math.random() * 10),
                'Field14': Math.floor(Math.random() * 10),
                'Field15': Math.floor(Math.random() * 1000),
                'Field16': Math.floor(Math.random() * 200),
                'Field17': Math.floor(Math.random() * 300),
            });
        }
    }
}

```

```

        'Field18': Math.floor(Math.random() * 400),
        'Field19': Math.floor(Math.random() * 500),
        'Field20': Math.floor(Math.random() * 700),
        'Field21': Math.floor(Math.random() * 800),
        'Field22': Math.floor(Math.random() * 1000),
        'Field23': Math.floor(Math.random() * 2000),
        'Field24': Math.floor(Math.random() * 150),
        'Field25': Math.floor(Math.random() * 1000),
        'Field26': Math.floor(Math.random() * 100),
        'Field27': Math.floor(Math.random() * 400),
        'Field28': Math.floor(Math.random() * 600),
        'Field29': Math.floor(Math.random() * 500),
        'Field30': Math.floor(Math.random() * 300),
    });
    }
}
</script>

```

COLUMN-VIRTUALIZATION.CS

```

public IActionResult Index()
{
    return View();
}

```

Note: Column's [Width](#) is required for column virtualization. If column's width is not defined then Grid will consider its value as 200px.

Virtualization with Grouping

Both the row and column virtualization can be used along with grouping. At initial rendering, the virtual height of scrollbar will be set based on the total number of records and after grouping, it will be refreshed based on the grouped state(expand/collapse). While collapse the group caption row in current viewport then the next view page grouped records are shown.

Note: The collapsed/expanded state will persist only for local dataSource while scrolling.

Limitations for virtual scrolling

- While using column virtual scrolling, column width should be in the pixel. Percentage values are not accepted.
- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited by the browser capability.
- The cell selection is not supported for both row and column virtual scrolling.
- Virtual scrolling is not compatible with batch editing, detail template, rowspan, colspan and hierarchy features.
- Group expand and collapse state will not be persisted.
- Since data is virtualized in grid, the aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the [Group with Page](#) topic.
- The page size provided must be two times larger than the number of visible rows in the grid. If the page size is failed to meet this condition then the size will be determined by grid.

- The height of the grid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

```
`css
.e-grid .e-row {
height: 2em;
}
`
```

- Programmatic selection using the `selectRows` method is not supported in virtual scrolling.

Browser height limitation in virtual scrolling and solution

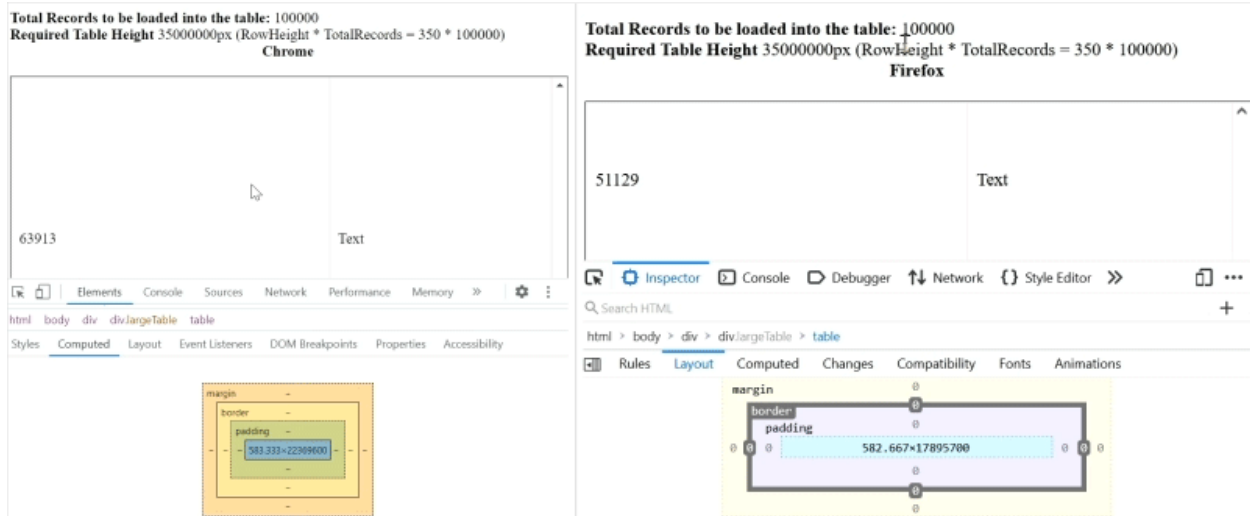
You can load millions of records in the Grid by using virtual scrolling, where the grid loads and renders rows on-demand while scrolling vertically. As a result, Grid lightens the browser's load by minimizing the DOM elements and rendering elements visible in the viewport. The height of the grid is calculated using the Total Records Count * [Row Height](#) property.

The browser has some maximum pixel height limitations for the scroll bar element. The content placed above the maximum height can't be scrolled if the element height is greater than the browser's maximum height limit. The browser height limit affects the virtual scrolling of the grid. When a large number of records are bound to the Grid, it can only display the records until the maximum height limit of the browser. Once the browser's height limit is reached while scrolling, the user won't be able to scroll further to view the remaining records.

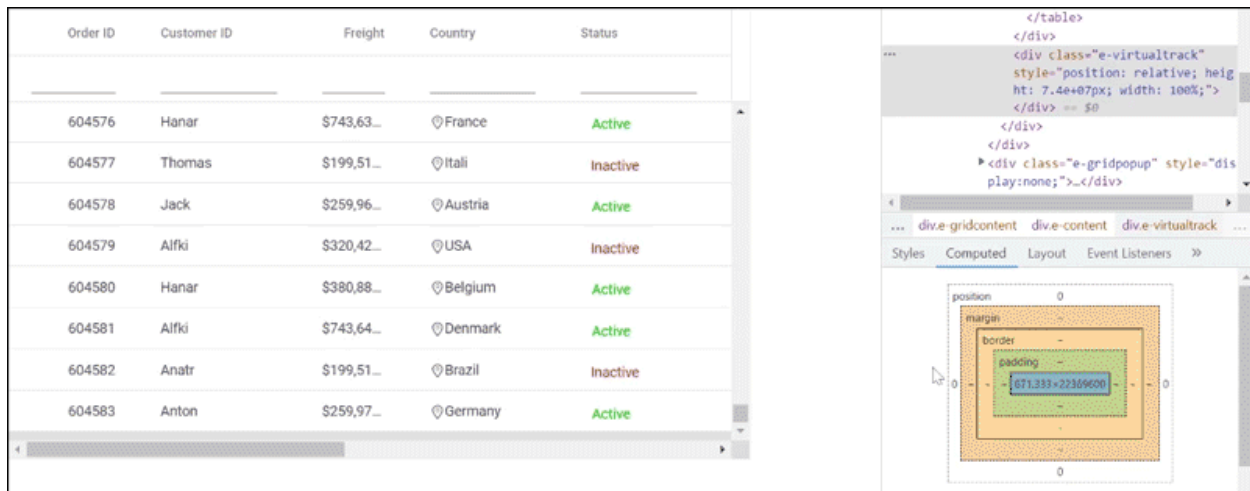
For example, if the row height is set as 30px and the total record count is 1000000 (1 million), then the height of the grid element will be 30,000,000 pixels. In this case, the browser's maximum height limit for a div is about 22,369,600 (The maximum pixel height limitation differs for different browsers). The records above the maximum height limit of the browser can't be scrolled.

This height limitation is not related to the Grid component. It fully depends on the default behavior of the browser. The same issue is reproduced in the normal HTML table too.

The following image illustrates the height limitation issue of a normal HTML table in different browsers (Chrome and Firefox).



Grid component also faced the same issue as mentioned in the below image.



The Grid has an option to overcome this limitation of the browser in the following ways.

[Solution 1: Using external buttons](#)

You can prevent the height limitation problem in the browser when scrolling through millions of records by loading the segment of data through different strategy.

In the following sample, Grid is rendered with a large number of records(nearly 2 million). Here, you can scroll 0.5 million records at a time in Grid. Once you reach the last page of 0.5 million records, the **Load Next Set** button will be shown at the bottom of the Grid. By clicking that button, you can view the next set of 0.5 million records in Grid. Also, the **Load Previous Set** button will be shown at the top of the Grid to load the previous set of 0.5 million records.

Let's see the step by step procedure for how we can overcome the limitation in the Syncfusion Grid component.

1. Create a custom adaptor by extending `UrlAdaptor` and binding it to the grid `DataSource` property. In the `processQuery` method of the custom adaptor, we handled the `Skip` query based on the current page set to perform the data operation with whole records on the server.

`csharp


```

export class CustomUrlAdaptor extends UrlAdaptor {
processQuery(args) {
if (arguments[1].queries) {
for (var i = 0; i < arguments[1].queries.length; i++) {
if (arguments[1].queries[i].fn === 'onPage') {
// pageSet - defines the number of segments that we are going to split the 2million records. In this
// example we have considered 0.5 million records for each set so the pageSet is 1, 2, 3 and 4.
// maxRecordsPerPageSet – In this example we define the value as 0.5 million.
// gridPageSize – the pageSize that we have defined in the Grid pageSettings->pageSize property
// customize the pageIndex based on the current pageSet (It send the skip query including the previous
// pageSet ) so that the other operations performed for total 2millions records instead of 0.5 million alone.
arguments[1].queries[i].e.pageIndex = (((pageSet - 1) * maxRecordsPerPageSet) / gridPageSize) +
arguments[1].queries[i].e.pageIndex;
}
}
}
var original = super.processQuery.apply(this, arguments);
return original;
}
}

this.dataManager = new DataManager({
adaptor: new CustomUrlAdaptor(),
url: "Home/UrlDatasource"
});
`

```

2.Render the grid by define the following features.

```

`csharp
<GridComponent id='grid' ref={g => this.grid = g} dataSource={this.dataManager}
enableVirtualization={true} pageSettings={this.pageSettings} height={360}
beforeDataBound={this.beforeDataBound}>
<ColumnsDirective>
.....
</ColumnsDirective>
</GridComponent>

```

3. In the `beforeDataBound` event, we set the `args.count` as 0.5 million to perform scrolling with 0.5 million records and all the data operations are performed with whole records which is handled using the custom adaptor. And also particular segment records count is less than 0.5 million means it will directly assigned the original segmented count instead of 0.5 million.

```
`csharp
beforeDataBound(args) {
// storing the total records count which means 2 million records count
totalRecords = args.count;
// change the count with respect to maxRecordsPerPageSet (maxRecordsPerPageSet = 500000)
args.count = args.count - ((pageSet - 1) * maxRecordsPerPageSet) > maxRecordsPerPageSet ?
maxRecordsPerPageSet : args.count - ((pageSet - 1) * maxRecordsPerPageSet);
}
```

4. Render “Load Next Set” button and “Load Previous Set” button at bottom and top of the grid component.

```
`csharp
<div className="pagearea1">
<ButtonComponent cssClass='e-info prevbtn' onClick={this.prevBtnClick} style="width:100%" >Load
Previous Set...</ButtonComponent>
</div>

<GridComponent id='grid' ref={g => this.grid = g} dataSource={this.dataManager}
enableVirtualization={true} pageSettings={this.pageSettings} height={360}
beforeDataBound={this.beforeDataBound} >
<ColumnsDirective>
.....
.....
</ColumnsDirective>
</GridComponent>
<div className="pagearea2">
<ButtonComponent cssClass='e-info nxtbtn' onClick={this.nxtBtnClick} style="width:100%" >Load Next
Set...</ButtonComponent>
</div>
```

5. While click on the **Load Next Set** / **Load Previous Set** button corresponding page data set is loaded to view remaining records of total 2 millions records after doing some simple calculation.

```

`typescript
// Triggered when clicking the Previous/ Next button.
prevNxtBtnClick(args) {
if (this.grid.element.querySelector('.e-content') && this.grid.element.querySelector('.e-
content').getAttribute('aria-busy') === 'false') {
// Increase/decrease the pageSet based on the target element.
pageSet = args.target.classList.contains('prevbtn') ? --pageSet : ++pageSet;
this.rerenderGrid(); // Re-render the Grid component.
}
}
`

```

You can find the full code sample from the below GitHub location.

Note: [View GitHub Sample](#)

Also, you can view the hosted link for this sample [here](#).

LOAD PREVIOUS SET...

Order ID	Customer ID	Freight	Country	Status
1	Alfki	\$12.30	📍Denmark	Active
2	Anatr	\$3.30	📍Brazil	Inactive
3	Anton	\$4.30	📍Germany	Active
4	Blomp	\$5.30	📍Austria	Inactive
5	Bolid	\$6.30	📍Switzerland	Active
6	Hanar	\$12.30	📍France	Active
7	Thomas	\$3.30	📍Itali	Inactive
8	Jack	\$4.30	📍Austria	Active
9	Alfki	\$5.30	📍USA	Inactive
10	Hanar	\$6.30	📍Belgium	Active

LOAD NEXT SET...

Note: If you perform grid actions such as filtering, sorting, etc., after scrolling through the 0.5 million data, the Grid performs those data actions with the whole records, not just the current loaded 0.5 million data.

Solution 2: Using RowHeight property

You can reduce the [row height](#) using the [RowHeight](#) property of the Grid. It will reduce the overall height to accommodate more rows. But this approach optimizes the limitation, but if the height limit is reached after reducing row height also, you have to opt for the previous solution or use paging.

In the following image, you can see how many records will be scrollable when setting rowHeight to "36px" and "30px".

RowHeight = 36px

Order ID	Customer ID	Freight	Country	Status
1	Alfki	\$12.30	📍Denmark	Active
2	Anatr	\$3.30	📍Brazil	Inactive
3	Anton	\$4.30	📍Germany	Active
4	Blomp	\$5.30	📍Austria	Inactive
5	Bolid	\$6.30	📍Switzerland	Active
6	Hanar	\$12.30	📍France	Active
7	Thomas	\$3.30	📍Itali	Inactive
8	Jack	\$4.30	📍Austria	Active

RowHeight = 30px

Order ID	Customer ID	Freight	Country	Status
1	Alfki	\$12.30	📍Denmark	Active
2	Anatr	\$3.30	📍Brazil	Inactive
3	Anton	\$4.30	📍Germany	Active
4	Blomp	\$5.30	📍Austria	Inactive
5	Bolid	\$6.30	📍Switzerland	Active
6	Hanar	\$12.30	📍France	Active
7	Thomas	\$3.30	📍Itali	Inactive
8	Jack	\$4.30	📍Austria	Active
9	Alfki	\$5.30	📍USA	Inactive
10	Hanar	\$6.30	📍Belgium	Active

Solution 3: Using paging instead of virtual scrolling

Similar to virtual scrolling, the [paging](#) feature also loads the data in an on-demand concept. Pagination is also compatible with all the other features (Grouping, Editing, etc.) in Grid. So, use the paging feature instead of virtual scrolling to view a large number of records in the Grid without any kind of performance degradation or browser height limitation.

- Programmatic selection using the `selectRows` method is not supported in virtual scrolling.

Infinite scrolling in ASP.NET MVC Grid Component

Infinite scrolling is used to load a huge amount of data without degrading the Grid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To enable Infinite scrolling, set `enableInfiniteScrolling` property as true.

Note: * In this feature, Grid will not make a new data request when you visit the same page again.

CSHTML

```
@Html.EJS().Grid("InfiniteGrid").DataBound("hide").EnableInfiniteScrolling()
.Height("500").Columns(col =>
{
    col.Field("Field1").HeaderText("PlayerName").Width("140").Add();

    col.Field("Field2").HeaderText("Year").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field3").HeaderText("Stint").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field4").HeaderText("TMID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field5").HeaderText("LGID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();
}).Render()

<script>
    var infiniteData = [], date1=null, date2=null, flag= true;
    var names = ['hardire', 'abramjo01', 'aubucch01', 'Hook',
'Rumpelstiltskin', 'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa',
'Anna', 'Snow White', 'Prince Charming', 'Cora', 'Zelena',
'August', 'Mulan', 'Graham', 'Discord', 'Will', 'Robin Hood',
'Jiminy Cricket', 'Henry', 'Neal', 'Red', 'Aaran', 'Aaren',
'Aarez', 'Aarman', 'Aaron', 'Aaron-James', 'Aarron', 'Aaryan', 'Aaryn',
'Aayan', 'Aazaan', 'Abaan', 'Abbas', 'Abdallah', 'Abdalroof',
'Abdihakim', 'Abdirahman', 'Abdisalam', 'Abdul', 'Abdul-Aziz',
'Abdulbasir', 'Abdulkadir', 'Abdulkarem', 'Abdulkhader',
'Abdullah', 'Abdul-Majeed', 'Abdulmalik', 'Abdul-Rehman', 'Abdur',
'Abdurraheem', 'Abdur-Rahman', 'Abdur-Rehmaan', 'Abel',
'Abhinav', 'Abhisumant', 'Abid', 'Abir', 'Abraham', 'Abu', 'Abubakar',
'Ace', 'Adain', 'Adam', 'Adam-James', 'Addison', 'Addisson',
'Adegbola', 'Adegbolahan', 'Aden', 'Adenn', 'Adie', 'Adil', 'Aditya',
'Adnan', 'Adrian', 'Adrien', 'Aedan', 'Aedin', 'Aedyn', 'Aeron',
'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-Aziz', 'Ahoua', 'Ahtasham',
```

```

        'Aiadan', 'Aidan', 'Aiden', 'Aiden-Jack', 'Aiden-Vee'];
        document.getElementById("generate").addEventListener('click', () =>
        {
            var grid =
            document.getElementById("InfiniteGrid").ej2_instances[0]
            if (!infiniteData.length) {
                show();
                dataSource();
                date1 = new Date().getTime();
                grid.dataSource = infiniteData;
            } else {
                flag = true;
                show();
                date1 = new Date().getTime();
                grid.refresh();
            }
        })
        function show() {
            document.getElementById('popup').style.display = 'inline-block';
        }
        function hide(args) {
            if (flag && date1) {
                var date2 = new Date().getTime();
                document.getElementById('performanceTime').innerHTML = 'Time
Taken: ' + (date2 - date1) + 'ms';
                flag = false;
            }
            document.getElementById('popup').style.display = 'none';
        }
        function dataSource() {
            for (var i= 0; i < 100000; i++) {
                infiniteData.push({
                    'Field1': names[Math.floor(Math.random() *
names.length)],
                    'Field2': 1967 + (i % 10),
                    'Field3': Math.floor(Math.random() * 200),
                    'Field4': Math.floor(Math.random() * 100),
                    'Field5': Math.floor(Math.random() * 2000),
                });
            }
        }
    }
</script>

```

INFINITE-SCROLL-NORMAL.CS

```

public IActionResult Index()
{
    return View();
}

```

InitialBlocks

You can define the initial loading pages count by using `infiniteScrollSettings.initialBlocks` property. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

CSHTML

```

@Html.EJS().Grid("InfiniteGrid").DataBound("hide").EnableInfiniteScrolling()
.InfiniteScrollSettings(settings => { settings.InitialBlocks(5);
}).Height("500").Columns(col =>
{
    col.Field("Field1").HeaderText("PlayerName").Width("140").Add();

col.Field("Field2").HeaderText("Year").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

col.Field("Field3").HeaderText("Stint").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

col.Field("Field4").HeaderText("TMID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

col.Field("Field5").HeaderText("LGID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();
}).Render()
<script>
    var infiniteData = [], date1=null, date2=null, flag= true;
    var names = ['hardire', 'abramjo01', 'aubucch01', 'Hook',
'Rumpelstiltskin', 'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa',
    'Anna', 'Snow White', 'Prince Charming', 'Cora', 'Zelena',
'August', 'Mulan', 'Graham', 'Discord', 'Will', 'Robin Hood',
    'Jiminy Cricket', 'Henry', 'Neal', 'Red', 'Aaran', 'Aaren',
'Aarez', 'Aarman', 'Aaron', 'Aaron-James', 'Aarron', 'Aaryan', 'Aaryn',
    'Aayan', 'Aazaan', 'Abaan', 'Abbas', 'Abdallah', 'Abdalroof',
'Abdihakim', 'Abdirahman', 'Abdisalam', 'Abdul', 'Abdul-Aziz',
    'Abdulbasir', 'Abdulkadir', 'Abdulkarem', 'Abdulkhader',
'Abdullah', 'Abdul-Majeed', 'Abdulmalik', 'Abdul-Rehman', 'Abdur',
    'Abdurraheem', 'Abdur-Rahman', 'Abdur-Rehmaan', 'Abel',
'Abhinav', 'Abhisumant', 'Abid', 'Abir', 'Abraham', 'Abu', 'Abubakar',
    'Ace', 'Adain', 'Adam', 'Adam-James', 'Addison', 'Addisson',
'Adegbola', 'Adegbolahan', 'Aden', 'Adenn', 'Adie', 'Adil', 'Aditya',
    'Adnan', 'Adrian', 'Adrien', 'Aedan', 'Aedin', 'Aedyn', 'Aeron',
'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-Aziz', 'Ahoua', 'Ahtasham',
    'Aiadan', 'Aidan', 'Aiden', 'Aiden-Jack', 'Aiden-Vee'];
    document.getElementById("generate").addEventListener('click', () =>
{
    var grid =
document.getElementById("InfiniteGrid").ej2_instances[0]
    if (!infiniteData.length) {
        show();
        dataSource();
        date1 = new Date().getTime();
        grid.dataSource = infiniteData;
    } else {
        flag = true;
        show();
        date1 = new Date().getTime();
        grid.refresh();
    }
})
function show() {

```



```

        document.getElementById('popup').style.display = 'inline-block';
    }
    function hide(args) {
        if (flag && datel) {
            var date2 = new Date().getTime();
            document.getElementById('performanceTime').innerHTML = 'Time
Taken: ' + (date2 - datel) + 'ms';
            flag = false;
        }
        document.getElementById('popup').style.display = 'none';
    }
    function dataSource() {
        for (var i= 0; i < 100000; i++) {
            infiniteData.push({
                'Field1': names[Math.floor(Math.random() *
names.length)],
                'Field2': 1967 + (i % 10),
                'Field3': Math.floor(Math.random() * 200),
                'Field4': Math.floor(Math.random() * 100),
                'Field5': Math.floor(Math.random() * 2000),
            });
        }
    }
}
</script>

```

INFINITE-SCROLL-INITIALBLOCKS.CS

```

public IActionResult Index()
{
    return View();
}

```

Cache Mode

Cache is used to store the loaded rows object in the Grid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the `infiniteScrollSettings.maxBlocks` count value, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set `infiniteScrollSettings.enableCache` property as true.

CSHTML

```

@Html.EJS().Grid("InfiniteGrid").DataBound("hide").EnableInfiniteScrolling()
.InfiniteScrollSettings(settings => { settings.EnableCache(true);
}).Height("500").Columns(col =>
{
    col.Field("Field1").HeaderText("PlayerName").Width("140").Add();

    col.Field("Field2").HeaderText("Year").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

    col.Field("Field3").HeaderText("Stint").Width("120").TextAlign(Syncfusion.EJ
2.Grids.TextAlign.Right).Add();
}

```

```

col.Field("Field4").HeaderText("TMID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();

col.Field("Field5").HeaderText("LGID").Width("120").TextAlign(Syncfusion.EJ2
.Grids.TextAlign.Right).Add();
}).Render()
<script>
    var infiniteData = [], date1=null, date2=null, flag= true;
    var names = ['hardire', 'abramjo01', 'aubucch01', 'Hook',
'Rumpelstiltskin', 'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa',
    'Anna', 'Snow White', 'Prince Charming', 'Cora', 'Zelena',
'August', 'Mulan', 'Graham', 'Discord', 'Will', 'Robin Hood',
    'Jiminy Cricket', 'Henry', 'Neal', 'Red', 'Aaran', 'Aaren',
'Aarez', 'Aarman', 'Aaron', 'Aaron-James', 'Aarron', 'Aaryan', 'Aaryn',
    'Aayan', 'Aazaan', 'Abaan', 'Abbas', 'Abdallah', 'Abdalroof',
'Abdihakim', 'Abdirahman', 'Abdisalam', 'Abdul', 'Abdul-Aziz',
    'Abdulbasir', 'Abdulkadir', 'Abdulkarem', 'Abdulkhader',
'Abdullah', 'Abdul-Majeed', 'Abdulmalik', 'Abdul-Rehman', 'Abdur',
    'Abdurraheem', 'Abdur-Rahman', 'Abdur-Rehmaan', 'Abel',
'Abhinav', 'Abhisumant', 'Abid', 'Abir', 'Abraham', 'Abu', 'Abubakar',
    'Ace', 'Adain', 'Adam', 'Adam-James', 'Addison', 'Addisson',
'Adegbola', 'Adegbolahan', 'Aden', 'Adenn', 'Adie', 'Adil', 'Aditya',
    'Adnan', 'Adrian', 'Adrien', 'Aedan', 'Aedin', 'Aedyn', 'Aeron',
'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-Aziz', 'Ahoua', 'Ahtasham',
    'Aiadan', 'Aidan', 'Aiden', 'Aiden-Jack', 'Aiden-Vee'];
    document.getElementById("generate").addEventListener('click', () =>
{
    var grid =
document.getElementById("InfiniteGrid").ej2_instances[0]
    if (!infiniteData.length) {
        show();
        dataSource();
        date1 = new Date().getTime();
        grid.dataSource = infiniteData;
    } else {
        flag = true;
        show();
        date1 = new Date().getTime();
        grid.refresh();
    }
})
function show() {
    document.getElementById('popup').style.display = 'inline-block';
}
function hide(args) {
    if (flag && date1) {
        var date2 = new Date().getTime();
        document.getElementById('performanceTime').innerHTML = 'Time
Taken: ' + (date2 - date1) + 'ms';
        flag = false;
    }
    document.getElementById('popup').style.display = 'none';
}
function dataSource() {
    for (var i= 0; i < 100000; i++) {
        infiniteData.push({

```

```

names.length)],
        'Field1': names[Math.floor(Math.random() *
        'Field2': 1967 + (i % 10),
        'Field3': Math.floor(Math.random() * 200),
        'Field4': Math.floor(Math.random() * 100),
        'Field5': Math.floor(Math.random() * 2000),
    });
    }
}
</script>

```

INFINITE-SCROLL-CACHE.CS

```

public IActionResult Index()
{
    return View();
}

```

Limitations for Infinite Scrolling

- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.
- Infinite scrolling is not compatible with batch editing, detail template and hierarchy features.
- The group records cannot be collapsed in cache mode.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the [Group with Page](#) topic.
- Programmatic selection using the [selectRows](#) and [selectRow](#) method is not supported in infinite scrolling.

Selection

Selection provides an option to highlight a row or a cell. It can be done through simple mouse down or arrow keys. To disable selection in the Grid, set the [AllowSelection](#) to false.

The grid supports two types of selection that can be set by using the [Type](#) property of [SelectionSettings](#). They are:

- **Single:** The Single value is set by default, and it only allows selection of a single row or a cell or a column.
- **Multiple:** Allows you to select multiple rows or cells or columns. To perform the multi-selection, press and hold CTRL key and click the desired rows or cells or columns. To select range of rows or cells or columns, press and hold the SHIFT key and click the rows or cells or columns.

CSHTML

```

@Html.EJS().Grid("SelectionAPI").DataSource((IEnumerable<object>) ViewBag.dataSource).AllowSelection().Columns(col =>
{
    col.Field("Inventor").HeaderText("Inventor Name").Width("180").Add();
}

```

```
col.Field("NumberofPatentFamilies").HeaderText("No of Patent
Families").Width("195").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add(
);

col.Field("Country").HeaderText("Country").Width("120").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
col.Field("Active").HeaderText("Active").Width("130").Add();
}).AllowPaging().SelectionSettings(select =>
select.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).PageSettings(page
=> page.PageCount(2)).Render()
```

SELECTION.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Selection mode

The grid supports three types of selection mode that can be set by using the [Mode](#) property of [SelectionSettings](#). They are:

- **Row:** The Row value is set by default, and allows you to select only rows.
- **Cell:** Allows you to select only cells.
- **Both:** Allows you to select rows and cells at the same time.

CSHTML

```
@Html.EJS().Grid("SelectionAPI").DataSource((IEnumerable<object>)ViewBag.dat
asource).AllowSelection().Columns(col =>
{
    col.Field("Inventor").HeaderText("Inventor Name").Width("180").Add();
    col.Field("NumberofPatentFamilies").HeaderText("No of Patent
Families").Width("195").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add(
);

col.Field("Country").HeaderText("Country").Width("120").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
col.Field("Active").HeaderText("Active").Width("130").Add();
}).AllowPaging().SelectionSettings(select =>
select.Mode(Syncfusion.EJ2.Grids.SelectionMode.Both)).PageSettings(page =>
page.PageCount(2)).Render()
```

SELECTION-MODE.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

```
}
```

Touch interaction

When you tap a grid row on touchscreen device, the tapped row is selected. It also shows a popup

![selection](images/selection.jpg)

for multi-row selection. To select multiple rows or cells, tap the popup

![mselection](images/mselection.jpg) and then tap the desired rows or cells.

Note: Multi-selection requires the selection **Type** to be **Multiple**.

The following screenshot represents a grid touch selection in the device.

![Touch interaction](images/touch-selection.jpg)

Aggregates

Aggregate values are displayed in the footer, group footer, or group caption of the Grid. It can be configured through [Aggregates](#) property. The [Field](#) and [Type](#) are the minimum properties required to represent an aggregate column.

By default, the aggregate value can be displayed in the footer, group, and caption cells. To show the aggregate value in one of the cells, use the [FooterTemplate](#), [GroupFooterTemplate](#), or [GroupCaptionTemplate](#) property.

Built-in aggregate types

The aggregate type should be specified in the [Type](#) property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- TrueCount
- FalseCount

Note: Multiple aggregates can be used for an aggregate column by setting the [Type](#) property with an array of aggregate types.

 Multiple types for a column is supported only when one of the aggregate templates is used.

Print in ASP.NET MVC Grid Component

To print the Grid, use the [print](#) method from grid instance. The print option can be displayed on the [Toolbar](#) by adding the **Print** toolbar item.

CSHTML

```
@Html.EJS().Grid("Print").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
```

```
col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().Toolbar(new List<string>() { "Print" }).Render()
```

PRINT.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Refer to the following links to know more about the browser page setup:

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

Print using an external button

To print the grid from an external button, invoke the [print](#) method.

CSHTML

```
@Html.EJS().Button("printbtn").Content("Print").IsPrimary(true).Render()
@Html.EJS().Grid("Print").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().Render()
<script>
    document.getElementById('printbtn').onclick = function () {
        document.getElementById('Print').ej2_instances[0].print();
    }
}
```

```
</script>
```

EXTERNAL-BTN.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Print the visible page

By default, the grid prints all the pages. To print the current page alone, set the [PrintMode](#) to **CurrentPage**.

CSHTML

```
@Html.EJS().Grid("Print").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().Toolbar(new List<string>() { "Print" })
.PrintMode(Syncfusion.EJ2.Grids.PrintMode.CurrentPage).Render()
```

CURRENT-PAGE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Print the hierarchy grid

By default, the grid will be print the master and expanded child grids alone. you can change the print option by using the [hierarchyPrintMode](#) property. The available options are,

- | | | |
|----------|---|--|
| Mode | Behavior | |
| ----- | ----- | |
| Expanded | Prints the master grid with expanded child grids. | |
| All | Prints the master grid with all the child grids. | |
| None | Prints the master grid alone. | |

CSHTML

```
@{
    var SecondChildGrid = new Syncfusion.EJ2.Grids.Grid() { DataSource =
        (IEnumerable<object>)ViewBag.CustomerDataSource,
        QueryString = "CustomerID",
        Columns = new List<Syncfusion.EJ2.Grids.GridColumn>
        {
            new Syncfusion.EJ2.Grids.GridColumn() { Field="CustomerID",
            HeaderText="Customer ID", Width="90",
            TextAlign=Syncfusion.EJ2.Grids.TextAlign.Right },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="ShipCountry",
            HeaderText="Ship Country", Width="90" },
        }
    };
    var ChildGrid = new Syncfusion.EJ2.Grids.Grid() { DataSource =
        (IEnumerable<object>)ViewBag.DataSource,
        QueryString = "EmployeeID",
        Columns = new List<Syncfusion.EJ2.Grids.GridColumn>
        {
            new Syncfusion.EJ2.Grids.GridColumn() { Field="OrderID",
            HeaderText="Order ID", Width="120" },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="Freight",
            HeaderText="Freight", Width="120", Format="C2",
            TextAlign=Syncfusion.EJ2.Grids.TextAlign.Right },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="ShipName",
            HeaderText="Ship Name", Width="150" },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="ShipCity",
            HeaderText="Ship City", Width="120" },
        },
        ChildGrid = SecondChildGrid
    };
}
@Html.EJS().Grid("HierarchyPrint").DataSource((IEnumerable<object>)ViewBag.EmpDataSource).HierarchyPrintMode(Syncfusion.EJ2.Grids.HierarchyGridPrintMode.All).Columns(col =>
{
    col.Field("EmployeeID").HeaderText("Employee ID").Width("125").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("FirstName").HeaderText("Name").Width("125").Add();
    col.Field("Country").HeaderText("Country").Width("180").Add();
    col.Field("City").HeaderText("City").Width("135").Add();
}).AllowSorting().Toolbar(new List<string>() {
    "Print"}).ChildGrid(ChildGrid).Render()
```

HIERARCHYPRINT.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrdersDetails.GetAllRecords();
    ViewBag.EmpDataSource = EmployeeView.GetAllRecords();
    ViewBag.CustomerDataSource = Customer.GetAllRecords();
    return View();
}
```

HIERARCHYPRINT.CS

```
public IActionResult Index()
```



```
{
    ViewBag.DataSource = OrdersDetails.GetAllRecords();
    ViewBag.EmpDataSource = EmployeeView.GetAllRecords();
    ViewBag.CustomerDataSource = Customer.GetAllRecords();
    return View();
}
```

Note: By default, the hierarchy grid prints the expanded child grids from the visible page only. Refer [To Print the expanded state grid from all pages.](#)

Print the master detail grid

The Grid has the option to visualize details of a record in another Grid in a master and detailed manner. By default, Grid will print the master grid alone. Instead of this, it is possible to print both the master and detail grids by using the [BeforePrint](#) event of the Grid.

In the following sample, the detail grid is added to the `element` argument of the `BeforePrint` event, resulting in both the master and detail grids being printed on the page.

CSHTML

```
<div>
    <B>Master Grid</B>

    @Html.EJS().Grid("masterGrid").DataSource((IEnumerable<object>)ViewBag.dataSource).Width("auto").SelectedRowIndex(0).Columns(col =>
    {

        col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Add();
        col.Field("OrderID").HeaderText("OrderID").Add();
        col.Field("CustomerID").HeaderText("FirstName").Add();
    }).RowSelected("selected").BeforePrint("beforePrint").Toolbar(new
    List<string>() { "Print" }).Render()
</div>
<p><div class='e-statustext'> Showing orders of Customer: <b
id=key></b></div></p>
<div>

    @Html.EJS().Grid("detailGrid").DataSource((IEnumerable<object>)ViewBag.dataSource).Columns(col =>
    {

        col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Add();

        col.Field("ShipCountry").HeaderText("ShipCountry").Add();
        col.Field("ShipName").HeaderText("ShipName").Add();
    }).Render()

</div>
<script>
    function selected(args) {
        var selectedRecord = args.data;
        var data = @Html.Raw(Json.Encode(ViewBag.dataSource));
        var employeeID = args.data.EmployeeID;
        var detaildata = new ej.data.DataManager(data).executeLocal(new
        ej.data.Query().where("EmployeeID", "equal", employeeID, false).take(10));
        var grid = document.getElementById("detailGrid").ej2_instances[0];
```

```

document.getElementById('key').innerHTML =
selectedRecord.EmployeeID;
grid.dataSource = new ej.data.DataManager(detaildata.slice(0,
5)).dataSource.json;
}
function beforePrint(args) {
    var customEle = document.createElement('div');
    var grid = document.getElementById("detailGrid").ej2_instances[0];
    customEle.innerHTML = document.getElementsByClassName('e-
statustext')[0].innerHTML + grid.element.innerHTML;
    customEle.appendChild(document.createElement('br'));
    args.element.append(customEle);
}
</script>

```

MASTER-DETAIL.CS

```

public IActionResult Index()
{
    ViewBag.dataSource = OrderDetails.GetAllRecords();
    return View();
}

```

Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence grid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.



Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the grid using [ToolBarClick](#) and [PrintComplete](#) events.

In the [ToolBarClick](#) event, based on **args.item.id** as **grid_print**. We can show or hide columns by setting [Visible](#) property of [Column](#) to **true** or **false** respectively.

In the [PrintComplete](#) event, We have reversed the state back to the previous state.

In the below example, we have **CustomerID** as a hidden column in the grid. While printing, we have changed **CustomerID** to visible column and **ShipCity** as hidden column.

CSHTML

```
@Html.EJS().Grid("Print").DataSource((IEnumerable<object>)ViewBag.DataSource)
.ToolbarClick("toolbarClick").PrintComplete("printComplete").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Visible(false).Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("150").Add();
}).AllowPaging().ToolBar(new List<string>() { "Print" }).Render()
<script>
    function toolbarClick(args) {
        for (var i = 0; i < this.columns.length; i++) {
            if (this.columns[i].field == "CustomerID") {
                this.columns[i].visible = true;
            }
            else if (this.columns[i].field == "ShipCity") {
                this.columns[i].visible = false;
            }
        }
    }
    function printComplete(args) {
        for (var i = 0; i < this.columns.length; i++) {
            if (this.columns[i].field == "CustomerID") {
                this.columns[i].visible = false;
            }
            else if (this.columns[i].field == "ShipCity") {
                this.columns[i].visible = true;
            }
        }
    }
}
</script>
```

SHOW-HIDE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
}
```

```
return View();
}
```

Limitations of Printing Large Data

When grid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang. Grid have option to handle large number of data by Virtualization. However while printing, it is not possible to use virtualization for rows and columns.

If printing of all the data is still needed, we suggest to Export the grid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

See Also

- [How to Print the expanded state grid from all pages](#)
- [How to print the Grid using external button in ASP.NET MVC Grid](#)
- [How to print the custom report in ASP.NET MVC Grid](#)

Adaptive View

The Grid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens. This interface will render the filter, sort, column chooser, column menu(supports only when the `rowRenderingMode` as Horizontal) column chooser, column menu(supports only when the `rowRenderingMode` as Horizontal) and edit dialogs adaptively and have an option to render the grid row elements in the vertical direction.

Render adaptive dialogs

When we enable the [enableAdaptiveUI](#) property, the grid will render the filter, sort, and edit dialogs in full screen for a better user experience. This behavior is demonstrated in the below demo.

CSHTML

```
<div class="e-bigger">
@Html.EJS().Grid("adaptivebrowser").DataSource((IEnumerable<object>)ViewBag.
DataSource).EnableAdaptiveUI(true).AllowPaging(true).AllowFiltering(true).Al
lowSorting(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("180").IsPrimaryKey(true).ValidationRules(new { required = true,
number = true }).Add();

    col.Field("Freight").HeaderText("Freight").Width("180").Format("C2").EditTyp
e("numericedit").ValidationRules(new { required = true }).Add();
    col.Field("CustomerID").HeaderText("Customer
ID").Width("180").ValidationRules(new { required = true, minLength = 3
}).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("180").Add();
}).FilterSettings(filter => {
filter.Type(Syncfusion.EJ2.Grids.FilterType.Excel); }).EditSettings(edit =>
{
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Dialog); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel", "Search" }).Render()
```

```
</div>
```

DEFAULT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Vertical row rendering

The grid will render the row elements in vertical order while setting the [rowRenderingMode](#) property value as **Vertical**.

CSHTML

```
<div class="e-bigger">
@Html.EJS().Grid("adaptivebrowser").DataSource((IEnumerable<object>)ViewBag.
DataSource).EnableAdaptiveUI(true).RowRenderingMode(Syncfusion.EJ2.Grids.Row
RenderingDirection.Vertical).AllowPaging(true).AllowFiltering(true).AllowSor
ting(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("180").IsPrimaryKey(true).ValidationRules(new { required = true,
number = true }).Add();

    col.Field("Freight").HeaderText("Freight").Width("180").Format("C2").EditTyp
e("numericedit").ValidationRules(new { required = true }).Add();
    col.Field("CustomerID").HeaderText("Customer
ID").Width("180").ValidationRules(new { required = true, minLength = 3
}).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("180").Add();
}).FilterSettings(filter => {
filter.Type(Syncfusion.EJ2.Grids.FilterType.Excel); }).EditSettings(edit =>
{
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Dialog); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel", "Search" }).Aggregates(agg =>
{
    agg.Columns(new List<Syncfusion.EJ2.Grids.GridAggregateColumn>() { new
Syncfusion.EJ2.Grids.GridAggregateColumn() { Field = "Freight", Format =
"C2", Type = "Sum", FooterTemplate = "Sum: ${Sum}" } }).Add();
}).Render()
</div>
```

DEFAULT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: * [enableAdaptiveUI](#) property must be enabled for vertical row rendering.

Supported features by vertical row rendering

The following features are only supported in vertical row rendering:

- Paging, including Page size dropdown
- Sorting
- Filtering
- Selection
- Dialog Editing
- Aggregate
- Infinite scroll
- Toolbar - Options like **Add**, **Filter**, **Sort**, **Edit**, **Delete**, **Search**, and **Toolbar template** are available when their respective features are enabled. The toolbar dynamically includes a three-dotted icon, containing additional features like **ColumnChooser**, **Print**, **PdfExport**, **ExcelExport**, or **CsvExport**, once these features are enabled. Please refer to the following snapshot.

![VerticalmodeColumnMenu](../images/VerticalmodeColumnMenu.gif)

A snapshot of the adaptive grid displaying enabled paging along with a pager dropdown.

![AdaptivePagerDropdown](../images/PagerDropdown_Adaptive.gif)

The Column Menu feature, which includes grouping, sorting, autofit, filter, and column chooser, is exclusively supported for the Grid in **Horizontal** [rowRenderingMode](#).

Scaffolding

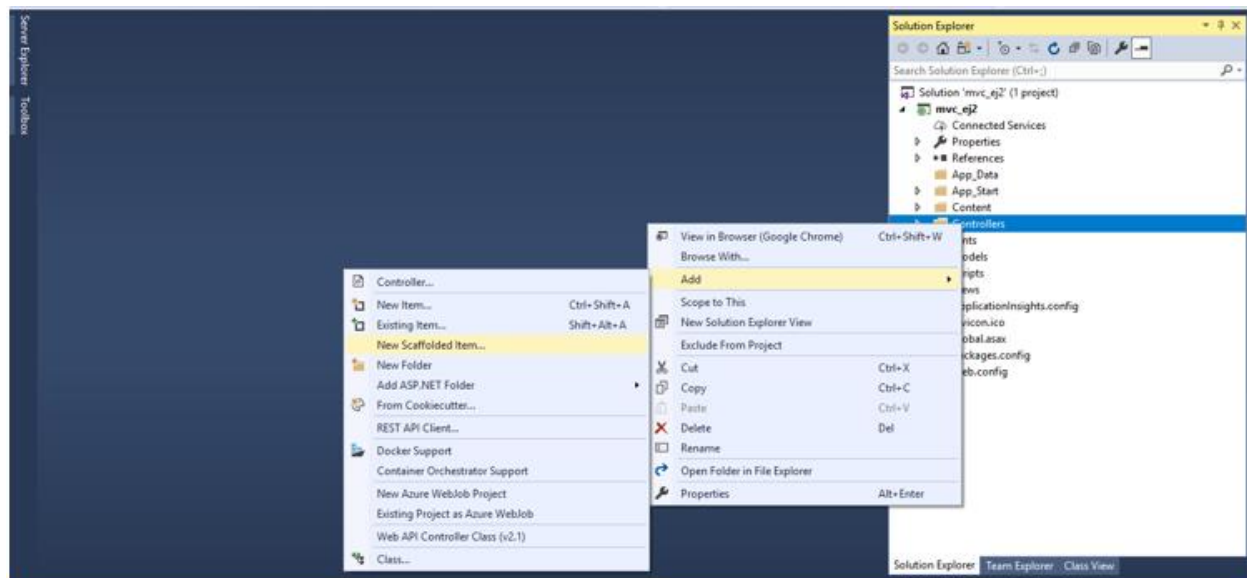
Syncfusion provides the **Visual Studio Scaffolding** for the Syncfusion ASP.NET MVC platform to quickly add code that interacts with data models and reduce the amount of time to develop with data operation in your project. Scaffolding provides an easier way to create Views and Controller action methods for Syncfusion ASP.NET MVC DataGrid controls.

Note: The Syncfusion ASP.NET MVC UI Scaffolder is available from **v16.4.0.40**.

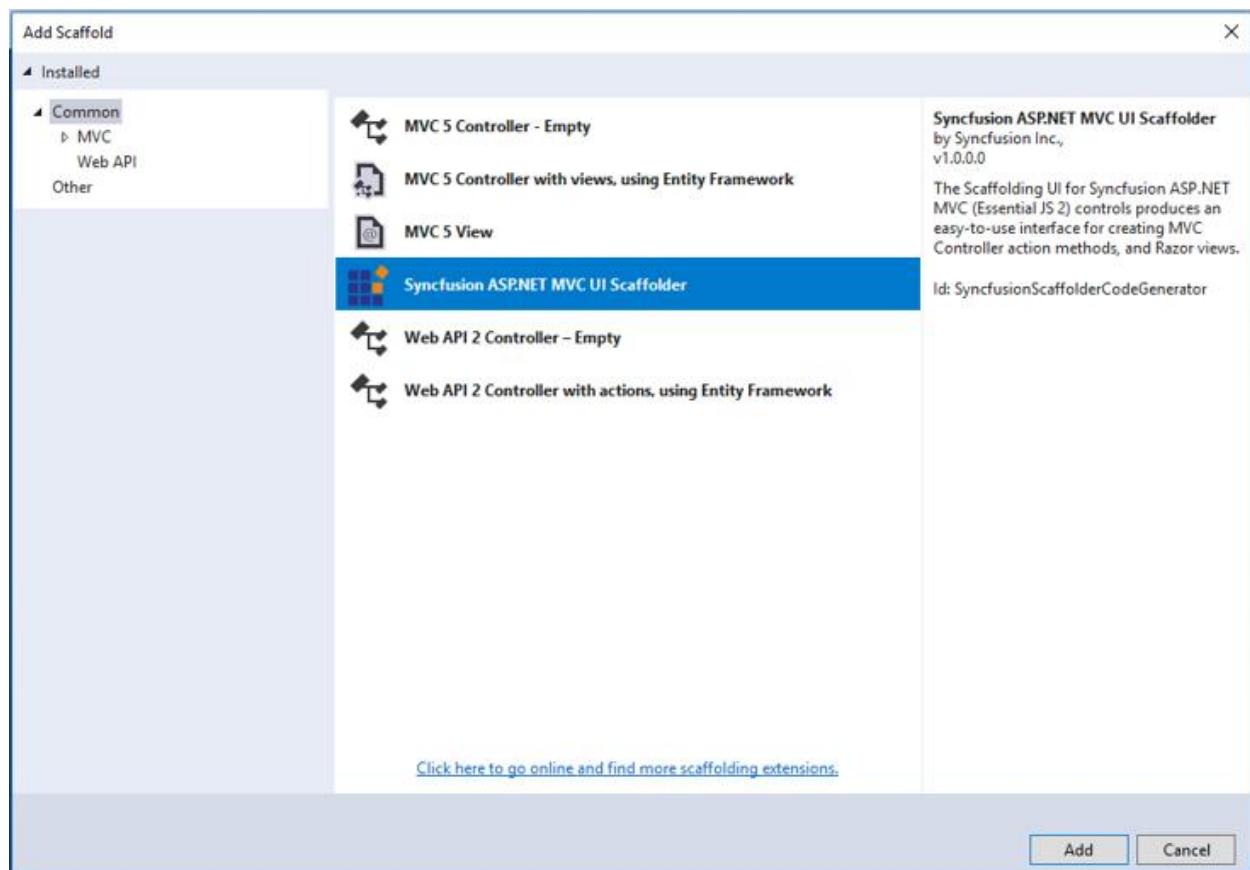
Add a Scaffolded Item

The following steps direct you to add a Scaffolded item to your ASP.NET MVC Web Application.

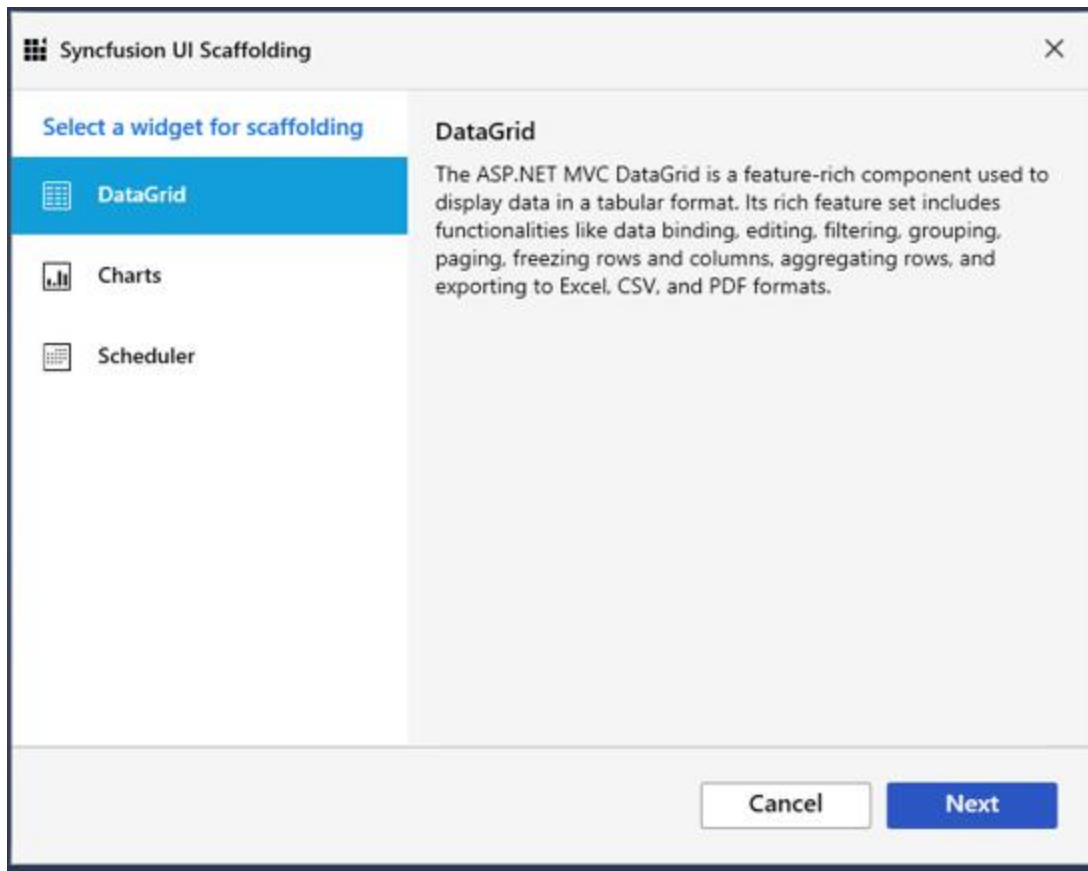
- Right-click on Controller folder in the Solution Explorer and select **Add → New Scaffolded Item**.



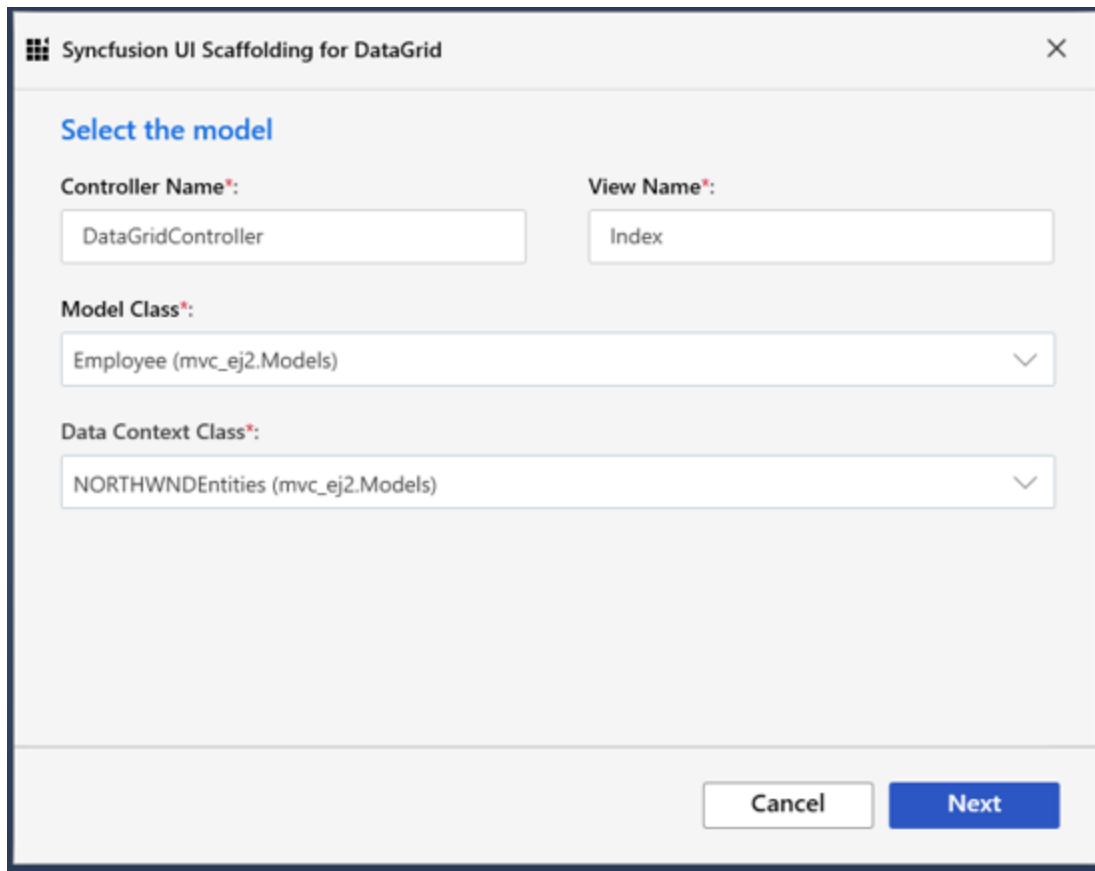
- You will see the Add Scaffold dialog. Select **Syncfusion ASP.NET MVC UI Scaffolder** and click 'Add' button, which will display the Syncfusion UI Scaffolding dialog.



- Select the DataGrid control to perform Scaffolding and click **Next**.



- Syncfusion UI Scaffolding for the selected control dialog will be opened. Enter the **Controller and View** names as per the application requirement. Also, select the required **Model Class** of the active project and its relevant **Data Context Class** and then Click **Next**.



Syncfusion UI Scaffolding for DataGrid

Select the model

Controller Name*: DataGridController

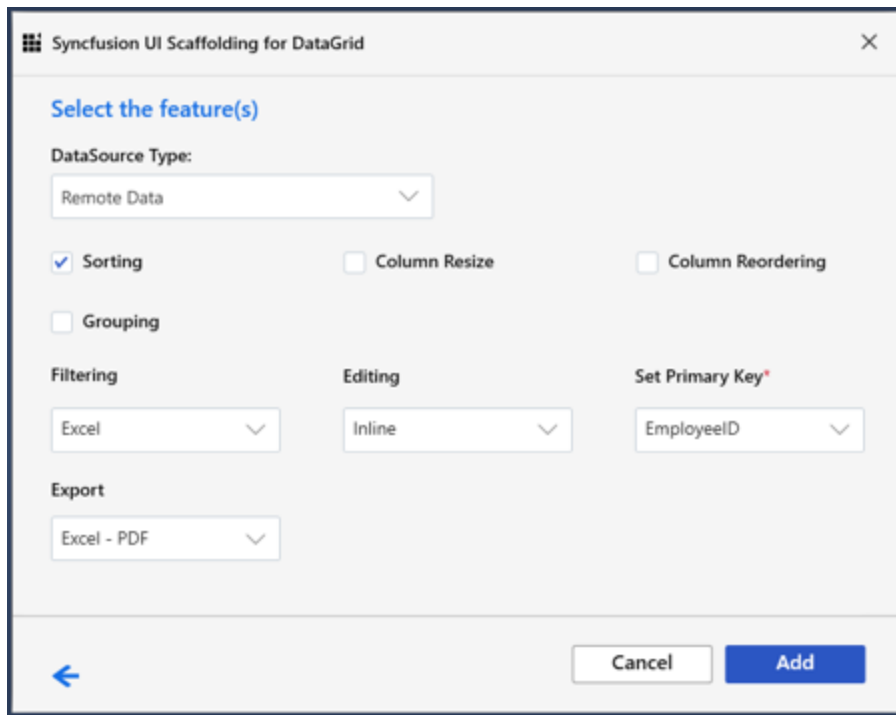
View Name*: Index

Model Class*: Employee (mvc_ej2.Models)

Data Context Class*: NORTHWNDEntities (mvc_ej2.Models)

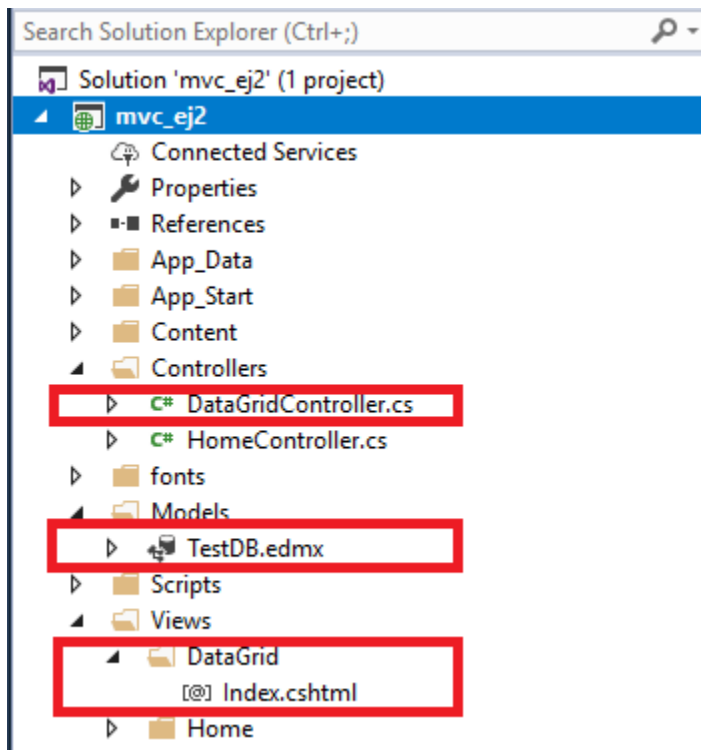
Cancel Next

- Syncfusion UI Scaffolding for the DataGrid control feature dialog will be opened. Select the required features and Click **Add**. Use the **back arrow** if you want to modify the already provided Controller or View name, and to change the **selected Model Class and Data Context Class**.



select the corresponding DataGrid features from option.

- The **Controller** and the corresponding **View** files are now generated with the selected features code snippet.



Note: * Ensure at least one Entity Framework model has been exists and the application compiled once.

 * If no Entity Framework model exists in your application, refer to the [documentation](#) to generate the Entity Framework model. Once model file added, ensure required DbContext and properties are added. Now, build the application and try scaffolding. If any changes done in model properties later, compile the application once before perform scaffolding.

How to render Syncfusion Control'?

Refer to the following UG links to render Syncfusion control after performed scaffolding.

MVC4: [Configure Essential JS 2 using Syncfusion.EJ2.MVC4 package](#)

MVC5: [Configure Essential JS 2 using Syncfusion.EJ2.MVC5 package](#)

State Persistence

State persistence refers to the Grid's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser. State persistence stores grid's model object in the local storage when the [EnablePersistence](#) is defined as true.

Restore initial Grid state

When the [EnablePersistence](#) property is set to **true**, the Grid will keep its state even if the page is reloaded. In some cases, you may be required to retain the Grid in its initial state. The Grid will not retain its initial state now since the [EnablePersistence](#) property has been enabled.

You can achieve this by destroying the grid after disabling the [EnablePersistence](#) property and clearing the local storage data, as shown in the sample below.

CSHTML

```
@Html.EJS().Button("restore").Content("Restore to initial
state").IsPrimary(true).Render()
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>) ViewBag.dataSource)
.AllowFiltering(true).EnablePersistence(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging(true).PageSettings(page => page.PageCount(5)).Render()
<script>
    document.getElementById('restore').onclick = function () {
        document.getElementById('Grid').ej2_instances[0].enablePersistence =
false;
        window.localStorage.setItem("gridGrid", "");
        document.getElementById('Grid').ej2_instances[0].destroy();
        //reloads the page
        location.reload();
    }
</script>
```

INITIAL-GRID.CS

```
public IActionResult Index()
```

```
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Maintaining custom query in a persistent state

The grid does not maintain the query params after page load event when the [EnablePersistence](#) is set to true. This is because the grid refreshes its query params for every page load. You can maintain the custom query params by resetting the `addParams` method in the [ActionBegin](#) event.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowFiltering(true).EnablePersistence(true).ActionBegin("actionBegin").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging().PageSettings(page => page.PageCount(5)).Render()

<script>
    function actionBegin(args) {
        this.query.addParams('$filter', 'EmployeeID eq 1');
    }
</script>
```

STATE-PERSIST.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Add a new column in persisted columns list

The Grid columns can be persisted when the [enablePersistence](#) property is set to true. If you want to add the new columns with the existing persist state, you can use the Grid inbuilt method such as `column.push` and call the `refreshColumns()` method for UI changes. Refer to the following sample for more information.

CSHTML

```
@Html.EJS().Button("add").Content("Add Columns").IsPrimary(true).Render()
```

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.EnablePersistence(true).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("170").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging(true).PageSettings(page => page.PageCount(5)).Render()
<script>
    document.getElementById('add').onclick = function () {
        var obj = { field: "Freight", headerText: 'Freight', width: 120 }
        document.getElementById('Grid').ej2_instances[0].columns.push(obj);
        //you can add the columns by using the Grid columns method
        document.getElementById('Grid').ej2_instances[0].refreshColumns();
    }
</script>
```

COLUMN-ADD.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.Datasource = orders;
    return View();
}
```

ToolBar in ASP.NET MVC Grid Component

The Grid provides ToolBar support to handle grid actions. The [ToolBar](#) property accepts either the collection of built-in toolbar items and **ItemModel** objects for custom toolbar items or HTML element ID for toolbar template.

Enable or disable toolbar items

You can enable/disable toolbar items by using the **enableItems** method.

CSHTML

```
@Html.EJS().Button("disable").Content("Disable").Render()
@Html.EJS().Button("enable").Content("Enable").Render()
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarClick("toolbarClick").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlig
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

col.Field("Freight").HeaderText("Freight").Width("120").EditType("numericedi
t").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
```

```

        col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").EditType("dropdownedit").Add();
    }).AllowPaging().PageSettings(page =>
page.PageCount(2)).AllowGrouping().GroupSettings(group => group.Columns(new
string[] { "CustomerID" })).ToolBar(new List<string>() { "Expand",
"Collapse" }).Render()
<script>
    document.getElementById("enable").addEventListener('click', () => {
        var grid = document.getElementById("Grid").ej2_instances[0];
        grid.toolbarModule.enableItems(['Grid_Collapse', 'Grid_Expand'],
true); // enable toolbar items.
    });
    document.getElementById("disable").addEventListener('click', () => {
        var grid = document.getElementById("Grid").ej2_instances[0];
        grid.toolbarModule.enableItems(['Grid_Collapse', 'Grid_Expand'],
false); // disable toolbar items.
    });
    function toolbarClick(args) {
        if (args.item.id === 'Grid_Collapse') { // grid_Collapse is
component id + '_' + toolbar item name.
            this.groupModule.collapseAll();
        }
        if (args.item.id === 'Grid_Expand') {
            this.groupModule.expandAll();
        }
    }
</script>

```

ENABLE-DISABLE.CS

```

public IActionResult Index()
{
    var orders = orderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}

```

Add toolbar at the bottom of Grid

You can add the Grid toolbar component at the bottom of Grid using the [Created](#) event.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("140").Add();

    col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("C").Width("120").Add();
    col.Field("OrderDate").HeaderText("Order
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width("140").Add();
}
)

```

```

}).Created("created").Toolbar(new List<string>() { "Print", "Search"
}).Render()
<script>
    function created() {
        let toolbar = this.element.querySelector('.e-toolbar');
        this.element.appendChild(toolbar);
    }
</script>

```

TOOLBAR-BOTTOM.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}

```

PDF Export

PDF export allows exporting Grid data to PDF document. You need to use the [PdfExport](#) method for exporting. To enable PDF export in the grid, set the [AllowPdfExport](#) as true.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().ToolbarClick("toolbarClick").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Toolbar(new List<string>() { "PdfExport" }).Render()
<script>
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        if (args.item.id === 'Grid_pdfexport') {
            gridObj.pdfExport();
        }
    }
</script>

```

PDF-EXPORT.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Show spinner while exporting

You can show/ hide spinner component while exporting the grid using **showSpinner/ hideSpinner** methods. You can use [ToolBarClick](#) event to show spinner before exporting and hide a spinner in the [PdfExportComplete](#) or [ExcelExportComplete](#) event after the exporting.

In the [ToolBarClick](#) event, based on the parameter **args.item.id** as **Gridpdfexport** or **Gridexcelexport** we can call the **showSpinner** method from grid instance.

In the [PdfExportComplete](#) or [ExcelExportComplete](#) event, We can call the **hideSpinner** method.

In the below demo, we have rendered the default spinner component when exporting the grid.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().AllowExcelExport().ToolBarClick("toolbarClick").PdfExportC
omplete("pdfExportComplete").ExcelExportComplete("excelExportComplete").Colu
mns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Visible(false).Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("ShipCity").HeaderText("ShipCity").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().ToolBar(new List<string>() { "PdfExport", "ExcelExport"
}).Render()
<script>
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        if (args.item.id === 'Grid_pdfexport') {
            gridObj.showSpinner();
            gridObj.pdfExport();
        }
        else if (args.item.id === 'Grid_excelexport') {
            gridObj.showSpinner();
            gridObj.excelExport();
        }
    }
    function pdfExportComplete(args) {
        this.hideSpinner();
    }
    function excelExportComplete(args) {
        this.hideSpinner();
    }
}
</script>
```

SHOW-SPINNER.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
```



```

    ViewBag.DataSource = Order;
    return View();
}

```

Custom data source

PDF export provides an option to define datasource dynamically before exporting. To export data dynamically, define the [dataSource](#) in [PdfExportProperties](#).

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().ToolBarClick("toolbarClick").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().ToolBar(new List<string>() { "PdfExport" }).Render()
<script>
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        if (args.item.id === 'Grid_pdfexport') {
            var data = [
                { OrderID: "100", CustomerID: "Vinet", Freight: "2.00",
OrderDate: new Date() },
                { OrderID: "101", CustomerID: "Hanar", Freight: "2.01",
OrderDate: new Date() },
                { OrderID: "102", CustomerID: "Mega", Freight: "4.48",
OrderDate: new Date() },
                { OrderID: "103", CustomerID: "Sam", Freight: "19.23",
OrderDate: new Date() }
            ];
            var pdfExportProperties = {
                dataSource: data
            };
            gridObj.pdfExport(pdfExportProperties);
        }
    }
</script>

```

CUSTOM-DATA.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Passing additional parameters to the server when exporting

You can pass the additional parameter in the [Query](#) property by invoking **addParams** method. In the [ToolbarClick](#) event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed **recordcount** as **12** using **addParams** method.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().AllowExcelExport().ToolbarClick("toolbarClick").PdfExportC
omplete("pdfExportComplete").ExcelExportComplete("excelExportComplete").Colu
mns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("ShipCity").HeaderText("ShipCity").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Toolbar(new List<string>() { "PdfExport", "ExcelExport"
}).Render()
<script>
    var queryClone;
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        if (args.item.id === 'Grid_pdfexport') {
            queryClone = gridObj.query;
            gridObj.query = new ej.data.Query().addParams("recordcount",
"12");
            gridObj.pdfExport();
        }
        else if (args.item.id === 'Grid_excelexport') {
            queryClone = gridObj.query;
            gridObj.query = new ej.data.Query().addParams("recordcount",
"12");
            gridObj.excelExport();
        }
    }
    function pdfExportComplete(args) {
        this.query = queryClone;
    }
    function excelExportComplete(args) {
        this.query = queryClone;
    }
</script>
```

ADDITIONAL-PARAMETER.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
```

```

    ViewBag.DataSource = Order;
    return View();
}

```

ADDITIONAL-PARAMETER.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Export large number of columns in a single page

By default, when the grid has large number of columns, then the PDF export will split more pages for the exceeded columns rendering. To achieve large number of columns exported in a single page, you need to set the **allowHorizontalOverflow** property as **false** in the [toolbarClick](#) event.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().ToolbarClick("toolbarClick").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Visible(false).Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("ShipCity").HeaderText("ShipCity").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();

    col.Field("EmployeeID").HeaderText("EmployeeID").Width("120").TextAlign(Sync
fusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("ShipCity").HeaderText("ShipCity").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();

    col.Field("ShipAddress").HeaderText("ShipAddress").Width("120").TextAlign(Sy
ncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("ShippedDate").HeaderText("ShippedDate").Width("120").TextAlign(Sy
ncfusion.EJ2.Grids.TextAlign.Right).Add();

    col.Field("ShipName").HeaderText("ShipName").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Toolbar(new List<string>() { "PdfExport" }).Render()
<script>
    function toolbarClick(args) {
        let pdfDocument = new ej.pdfexport.PdfDocument();
        let page = pdfDocument.pages.add();
        var grid = document.getElementById('Grid').ej2_instances[0];
    }

```

```

var pdfGrid = new ej.pdfexport.PdfGrid();
pdfGrid.style.allowHorizontalOverflow = false;
let format = new ej.pdfexport.PdfLayoutFormat();
var collength = grid.columns.length;
pdfGrid.columns.add(collength);
pdfGrid.headers.add(1);
var pdfGridHeader = pdfGrid.headers.getHeader(0);
for (let j = 0; j < collength; j++) {
    pdfGridHeader.cells.getCell(j).value =
grid.columns[j].headerText;
}
let pdfGridRow1 = [];
for (let k = 0; k < grid.dataSource.length; k++) {
    pdfGridRow1[k] = pdfGrid.rows.addRow();
}
for (let k = 0; k < grid.dataSource.length; k++) {
    for (let r = 0; r < grid.columnModel.length; r++) {
        pdfGridRow1[k].cells.getCell(r).value =
grid.dataSource[k][grid.columnModel[r].field].toString();
    }
}
pdfGrid.draw(page, new ej.pdfexport.PointF(0, 0), format);
pdfDocument.save('output.pdf');
pdfDocument.destroy();
}
</script>

```

LARGE-COLUMNS.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

See Also

- [How can I change a layout to export to pdf in ASP.NET MVC Grid](#)

Excel Export

The excel export allows exporting Grid data to Excel document. You need to use the [excelExport](#) method for exporting. To enable Excel export in the grid, set the [AllowExcelExport](#) property as true.

To use excel export, You need to define the **ExcelExport** in inbuild toolbar and define the [ToolbarClick](#) event for exporting the Grid.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowExcelExport().ToolbarClick("toolbarClick").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}
)

```

```

        col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
        col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
    }).AllowPaging().Toolbar(new List<string>() { "ExcelExport" }).Render()
<script>
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        if (args.item.id === 'Grid_excelexport') {
            gridObj.excelExport();
        }
    }
</script>

```

EXCEL-EXPORT.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Show spinner while exporting

You can show/ hide spinner component while exporting the grid using **showSpinner/ hideSpinner** methods. You can use [ToolbarClick](#) event to show spinner before exporting and hide a spinner in the [PdfExportComplete](#) or [ExcelExportComplete](#) event after the exporting.

In the [ToolbarClick](#) event, based on the parameter **args.item.id** as **Gridpdfexport or Gridexcelexport** we can call the **showSpinner** method from grid instance.

In the [PdfExportComplete](#) or [ExcelExportComplete](#) event, We can call the **hideSpinner** method.

In the below demo, we have rendered the default spinner component when exporting the grid.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().AllowExcelExport().ToolbarClick("toolbarClick").PdfExportC
omplete("pdfExportComplete").ExcelExportComplete("excelExportComplete").Colu
mns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Visible(false).Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("ShipCity").HeaderText("ShipCity").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();
}

```

```

        col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
    }).AllowPaging().Toolbar(new List<string>() { "PdfExport", "ExcelExport"
    }).Render()
<script>
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];
        if (args.item.id === 'Grid_pdfexport') {
            gridObj.showSpinner();
            gridObj.pdfExport();
        }
        else if (args.item.id === 'Grid_excelexport') {
            gridObj.showSpinner();
            gridObj.excelExport();
        }
    }
    function pdfExportComplete(args) {
        this.hideSpinner();
    }
    function excelExportComplete(args) {
        this.hideSpinner();
    }
</script>

```

SHOW-SPINNER.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Custom data source

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the [dataSource](#) in [ExcelExportProperties](#).

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowExcelExport().ToolbarClick("toolbarClick").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Toolbar(new List<string>() { "ExcelExport" }).Render()
<script>
    function toolbarClick(args) {
        var gridObj = document.getElementById("Grid").ej2_instances[0];

```

```

        if (args.item.id === 'Grid_excelexport') {
            var data = [
                { OrderID: "100", CustomerID: "Vinet", Freight: "2.00",
OrderDate: new Date() },
                { OrderID: "101", CustomerID: "Hanar", Freight: "2.01",
OrderDate: new Date() },
                { OrderID: "102", CustomerID: "Mega", Freight: "4.48",
OrderDate: new Date() },
                { OrderID: "103", CustomerID: "Sam", Freight: "19.23",
OrderDate: new Date() }
            ];
            var excelExportProperties = {
                dataSource: data
            };
            gridObj.excelExport(excelExportProperties);
        }
    }
</script>

```

CUSTOM-DATA.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Passing additional parameters to the server when exporting

You can pass the additional parameter in the [Query](#) property by invoking **addParams** method. In the [ToolbarClick](#) event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed **recordcount** as **12** using **addParams** method.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowPdfExport().AllowExcelExport().ToolbarClick("toolbarClick").PdfExportC
omplete("pdfExportComplete").ExcelExportComplete("excelExportComplete").Colu
mns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("ShipCity").HeaderText("ShipCity").Width("120").TextAlign(Syncfusi
on.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().Toolbar(new List<string>() { "PdfExport", "ExcelExport"
}).Render()
<script>

```

```

var queryClone;
function toolbarClick(args) {
    var gridObj = document.getElementById("Grid").ej2_instances[0];
    if (args.item.id === 'Grid_pdfexport') {
        queryClone = gridObj.query;
        gridObj.query = new ej.data.Query().addParams("recordcount",
"12");
        gridObj.pdfExport();
    }
    else if (args.item.id === 'Grid_excelexport') {
        queryClone = gridObj.query;
        gridObj.query = new ej.data.Query().addParams("recordcount",
"12");
        gridObj.excelExport();
    }
}
function pdfExportComplete(args) {
    this.query = queryClone;
}
function excelExportComplete(args) {
    this.query = queryClone;
}
</script>

```

ADDITIONAL-PARAMETER.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

ADDITIONAL-PARAMETER.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

See Also

- [Copy & paste excel](#)
- [Copy and pasting excel sheet data to Grid](#)

Globalization in ASP.NET MVC Grid control

Localization

The [Localization](#) library allows you to localize default text content of the Grid. The grid component has static text on some features (like group drop area text, pager information text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [Locale](#) value and translation object.

The following list of properties and its values are used in the grid.

|Locale keywords|Text|

|----|----|

|EmptyRecord |No records to display.|

|True |true|

|False |false|

|InvalidFilterMessage |Invalid filter data.|

|GroupDropArea |Drag a column header here to group its column.|

|UnGroup |Click here to ungroup.|

|GroupDisable |Grouping is disabled for this column.|

|FilterbarTitle |\s filter bar cell.|

|EmptyDataSourceError |DataSource must not be empty at initial load as columns are generated from the dataSource in AutoGenerate Column Grid.|

|Add | Add|

|Edit| Edit|

|Cancel| Cancel|

|Update| Update|

|Delete | Delete|

|Print | Print|

|Pdfexport | PDF Export|

|Excelexport | Excel Export|

|Wordexport | Word Export|

|Csvexport | CSV Export|

|Search | Search|

|Columnchooser | Columns|

|Save | Save|

|Item | item|

|Items | items|

|EditOperationAlert | No records selected for edit operation|

|DeleteOperationAlert | No records selected for delete operation|

|SaveButton | Save|

|OKButton | OK|

|CancelButton | Cancel|

EditFormTitle	Details of
AddFormTitle	Add New Record
BatchSaveConfirm	Are you sure you want to save changes?
BatchSaveLostChanges	Unsaved changes will be lost. Are you sure you want to continue?
ConfirmDelete	Are you sure you want to Delete Record?
CancelEdit	Are you sure you want to Cancel the changes?
ChooseColumns	Choose Column
SearchColumns	search columns
Matchs	No Matches Found
FilterButton	Filter
ClearButton	Clear
StartsWith	Starts With
EndsWith	Ends With
Contains	Contains
Equal	Equal
NotEqual	Not Equal
LessThan	Less Than
LessThanOrEqual	Less Than Or Equal
GreaterThan	Greater Than
GreaterThanOrEqual	Greater Than Or Equal
ChooseDate	Choose a Date
EnterValue	Enter the value
Copy	Copy
Group	Group by this column
Ungroup	Ungroup by this column
autoFitAll	AutoFit all columns
autoFit	AutoFit this column
Export	Export
FirstPage	First Page
LastPage	Last Page
PreviousPage	Previous Page
NextPage	Next Page
SortAscending	Sort Ascending

SortDescending	Sort Descending
EditRecord	Edit Record
DeleteRecord	Delete Record
FilterMenu	Filter
SelectAll	Select All
Blanks	Blanks
FilterTrue	True
FilterFalse	False
NoResult	No Matches Found
ClearFilter	Clear Filter
NumberFilter	Number Filters
TextFilter	Text Filters
DateFilter	Date Filters
MatchCase	Match Case
Between	Between
CustomFilter	Custom Filter
CustomFilterPlaceholder	Enter the value
CustomFilterDatePlaceholder	Choose a date
AND	AND
OR	OR
ShowRowsWhere	Show rows where
currentPageInfo	{0} of {1} pages
totalItemsInfo	({0} items)
firstPageTooltip	Go to first page
lastPageTooltip	Go to last page
nextPageTooltip	Go to next page
previousPageTooltip	Go to previous page
nextPagerTooltip	Go to next pager items
previousPagerTooltip	Go to previous pager items
pagerDropDown	Items per page
pagerAllDropDown	Items
All	All

Loading translations

To load translation object in an application, use **load** function of the **L10n** class.

The following example demonstrates the Grid in **Deutsch** culture.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().AllowGrouping().Locale("de-DE").PageSettings(page =>
page.PageSize(6)).Render()
<script>
    ej.base.L10n.load({
        'de-DE': {
            'grid': {
                'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
                'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
                'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
                'EmptyDataSourceError': 'DataSource darf bei der
Erstauslastung nicht leer sein, da Spalten aus der dataSource im
AutoGenerate Spaltenraster',
                'Item': 'Artikel',
                'Items': 'Artikel'
            },
            'pager': {
                'currentPageInfo': '{0} von {1} Seiten',
                'totalItemsInfo': '({0} Beiträge)',
                'firstPageTooltip': 'Zur ersten Seite',
                'lastPageTooltip': 'Zur letzten Seite',
                'nextPageTooltip': 'Zur nächsten Seite',
                'previousPageTooltip': 'Zurück zur letzten Seit',
                'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-
Elementen',
                'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
            }
        }
    });
</script>
```

LOCALIZATION.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in grid component using format strings in the Format.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("Freight").HeaderText("Ship
Name").Format("C2").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().AllowGrouping().Locale("de-DE").PageSettings(page =>
page.PageSize(6)).Render()
<script>
    ej.base.setCurrencyCode('EUR');
    ej.base.L10n.load({
        'de-DE': {
            'grid': {
                'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
                'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
                'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
                'EmptyDataSourceError': 'DataSource darf bei der
Erstauslastung nicht leer sein, da Spalten aus der dataSource im
AutoGenerate Spaltenraster',
                'Item': 'Artikel',
                'Items': 'Artikel'
            },
            'pager': {
                'currentPageInfo': '{0} von {1} Seiten',
                'totalItemsInfo': '({0} Beiträge)',
                'firstPageTooltip': 'Zur ersten Seite',
                'lastPageTooltip': 'Zur letzten Seite',
                'nextPageTooltip': 'Zur nächsten Seite',
                'previousPageTooltip': 'Zurück zur letzten Seit',
                'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-
Elementen',
                'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
            }
        }
    });
</script>
```

INTERNATIONALIZATION.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

```
}

```

Note: In the above sample, **Freight** column is formatted by **NumberFormatOptions**.

By default, [Locale](#) value is **en-US**. If you want to change the **en-US** culture to a different culture, you have to change the [Locale](#) accordingly.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Grid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the [EnableRtl](#) to true.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().AllowGrouping().EnableRtl(true).Locale("de-
DE").PageSettings(page => page.PageSize(6)).Render()
<script>
    ej.base.L10n.load({
        'de-DE': {
            'grid': {
                'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
                'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
                'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
                'EmptyDataSourceError': 'DataSource darf bei der
Erstauslastung nicht leer sein, da Spalten aus der dataSource im
AutoGenerate Spaltenraster',
                'Item': 'Artikel',
                'Items': 'Artikel'
            },
            'pager': {
                'currentPageInfo': '{0} von {1} Seiten',
                'totalItemsInfo': '({0} Beiträge)',
                'firstPageTooltip': 'Zur ersten Seite',
                'lastPageTooltip': 'Zur letzten Seite',
                'nextPageTooltip': 'Zur nächsten Seite',
                'previousPageTooltip': 'Zurück zur letzten Seit',
                'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-
Elementen',
                'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
            }
        }
    });
</script>
```

RIGHTTOLEFT.CS

```
public IActionResult Index()
{
    var orders = OrderDetails.GetAllRecords();
    ViewBag.datasource = orders;
    return View();
}
```

See Also

- [Internationalization](#)
- [Localization](#)

Accessibility in Grid component

The Grid component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

WAI-ARIA attributes

The Grid component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Grid component:

Attributes	Purpose
---	---
<code>role=grid</code>	To represent the element containing the grid component.
<code>role=row</code>	To represent the element containing the cells of the row in the grid.
<code>role=rowgroup</code>	To represent the group of rows in the grid.
<code>role=columnheader</code>	To represent the cell in a row contains header information for a column in the grid.
<code>role=gridcell</code>	To represent a cell in the grid component.
<code>role=button</code>	To represent the element that acts as a button in the grid.
<code>role=search</code>	To represent the element that acts as a search region in the grid.
<code>role=presentation</code>	To represent the element to be not available for accessibility concerns.
<code>role=navigation</code>	To represent the element containing pager elements to navigate from one page to another.
<code>aria-colindex</code>	Defines the column index of the column with respect to the total number of columns within the grid.
<code>aria-rowindex</code>	Defines row index of the row with respect to the total number of rows within the grid.
<code>aria-rowspan</code>	Defines the number of rows spanned by a cell within the grid.
<code>aria-colspan</code>	Defines the number of columns spanned by a cell within the grid.
<code>aria-rowcount</code>	Defines the total number of rows in the grid.

- | **aria-colcount** | Defines the total number of columns in the grid. |
- | **aria-selected** | Indicates the current "selected" state of the rows and cells in the grid. |
- | **aria-expanded** | Indicate if the expand icon in the hierarchy grid or grouped grid or detail grid is expanded or collapsed |
- | **aria-sort** | Indicates whether the data in the grid are sorted in ascending or descending order. |
- | **aria-busy** | Indicates an element is being modified and that assistive technologies may want to wait until the changes are complete before informing the user about the update. |
- | **aria-owns** | Identifies an element in order to define a visual, functional, or contextual relationship between a parent and its child elements. |
- | **aria-hidden** | Hides the element from accessibility concerns. |
- | **aria-labelledby** | Provides an accessible name for the checkbox labels in excel filter, checkbox filter and column chooser dialog. |
- | **aria-describedby** | Provides an description about the features enabled in the header when the grid header cell is focused. |

Keyboard interaction

The Grid component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Grid component.

Pager

Windows | MAC | To do this

Home | Fn + Left Arrow | Moves the focus to the first cell of the focused row.

End | Fn + Right Arrow | Moves the focus to the last cell of the focused row.

Ctrl + Home | Command + Fn + Left Arrow | Moves the focus to the first Cell of the first row in the grid.

Ctrl + End | Command + Fn + Right Arrow | Moves the focus to the last Cell of the last row in the grid.

Up Arrow | Up Arrow | Moves the cell focus upward from the focused cell.

Down Arrow | Down Arrow | Moves the cell focus downward from the focused cell.

Right Arrow | Right Arrow | Moves the cell focus right side from the focused cell.

Left Arrow | Left Arrow | Moves the cell focus left side from the focused cell.

Alt + J | Alt + J | Moves the focus to the entire grid.

Alt + W | Alt + W | Move the focus to the grid content element.

Selection

Windows | MAC | To do this

Ctrl + Up Arrow | Command + Up Arrow | Collapses all the visible groups.

Ctrl + Down Arrow | Command + Down Arrow | Expands all the visible groups.

Ctrl + Space | Ctrl + Space | Performs grouping when focused on a header element.

Enter | Enter | If the current cell is an expand/collapse cell then expands/collapses the current group/detailrow/childgrid.

Print

Windows | MAC | To do this

Ctrl + C | Command + C | Copies selected rows or cells data into the clipboard.

Ctrl + Shift + H | Ctrl + Shift + H | Copies selected rows or cells data with header into clipboard

Editing

Windows | MAC | To do this

Alt + Down arrow | Alt + Down arrow | Opens the filter menu(excel, menu and checkbox filter) when its header element is in focused state.

Column Menu

Windows | MAC | To do this

Ctrl + left arrow or right arrow | Command + left arrow or right arrow | Reorders the focused header column to the left or right side.

Sorting

Windows | MAC | To do this

Enter | Enter | Performs sorting(ascending/descending) on a column when its header element is in focused state.

Ctrl + Enter | Command + Enter | Performs multi-sorting on a column when its header element is in focused state.

Shift + Enter | Shift + Enter | Clears sorting for the focused header column.

*** The Command and Control keys on Mac devices can be interchanged. When this switch occurs, use the Command key in place of the Control key and the Control key in place of the Command key for the above listed key interactions with Mac devices.**

*** For example, after switching the keys to group the columns when the header element is focused use Command + Space and for expanding the visible groups use Ctrl + Down Arrow.**

Ensuring accessibility

The Grid component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Grid component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Grid component with accessibility tools.

See also

- [Accessibility in Syncfusion Grid control](#)

Clipboard

The clipboard provides an option to copy selected rows or cells data into the clipboard.

The following list of keyboard shortcuts is supported in the Grid to copy selected rows or cells data into the clipboard.

Interaction keys	Description
----- -----	
Ctrl + C	Copy selected rows or cells data into clipboard.
Ctrl + Shift + H	Copy selected rows or cells data with header into clipboard.

CSHTML

```
@(Html.EJS().Grid("Clipboard").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("C2").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().PageSettings(new Syncfusion.EJ2.Grids.GridPageSettings() { PageCount = 5 }).Render())
```

LOCAL.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Copy to clipboard by external buttons

To copy selected rows or cells data into the clipboard with help of toolbar buttons, you need to define the [ToolbarClick](#) event and invoke the [copy](#) method.

CSHTML

```
@Html.EJS().Button("Copy").Content("Copy").IsPrimary(true).Render()
@Html.EJS().Button("CopyHeader").Content("Copy With Header").IsPrimary(true).Render()
@Html.EJS().Grid("CopyBtn").DataSource((IEnumerable<object>)ViewBag.DataSource).Columns(col =>
```

```

{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.Te
xtAlign.Right).Width("120").Format("C2").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().Render()
<script>
    document.getElementById("Copy").onclick = function () {
        var grid = document.getElementById("CopyBtn").ej2_instances[0];
        grid.copy();
    }
    document.getElementById("CopyHeader").onclick = function () {
        var grid = document.getElementById("CopyBtn").ej2_instances[0];
        grid.copy(true);
    }
</script>

```

LOCAL.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

AutoFill

AutoFill Feature allows you to copy the data of selected cells and paste it to another cells by just dragging the autofill icon of the selected cells up to required cells. This feature is enabled by defining [EnableAutoFill](#) property as true.

CSHTML

```

@Html.EJS().Grid("Autofill").DataSource((IEnumerable<object>)ViewBag.DataSou
rce).AllowSelection().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").ValidationRules(new { required =
"true"}).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Width("150").ValidationRules(new { required = "true", minLength=3
}).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().PageSettings(page =>
page.PageCount(2)).EnableAutoFill().SelectionSettings(select =>

```

```
select.CellSelectionMode(Syncfusion.EJ2.Grids.CellSelectionMode.Box).Mode(Syncfusion.EJ2.Grids.SelectionMode.Cell).Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).EditSettings(edit => {
    edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Batch); }).Toolbar(new List<string>() { "Add", "Update", "Cancel" }).Render()
```

AUTOFILL.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: If [EnableAutoFill](#) is set to true, then the autofill icon will be displayed on cell selection to copy cells.

 It requires the selection [Mode](#) to be **Cell**, [CellSelectionMode](#) to be **Box** and also Batch Editing should be enabled.

Limitations of AutoFill

- Since the string values are not parsed to number and date type, so when the selected string type cells are dragged to number type cells then it will display as **NaN**. For date type cells, when the selected string type cells are dragged to date type cells then it will display as an **empty cell**.
- Linear series and the sequential data generations are not supported in this autofill feature.

Paste

You can able to copy the content of a cell or a group of cells by selecting the cells and pressing Ctrl + C shortcut key and paste it to another set of cells by selecting the cells and pressing Ctrl + V shortcut key.

CSHTML

```
@Html.EJS().Grid("Paste").DataSource((IEnumerable<object>)ViewBag.DataSource).AllowSelection().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Width("120").ValidationRules(new { required = "true" }).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").ValidationRules(new { required = "true", minLength=3 }).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().PageSettings(page =>
page.PageCount(2)).SelectionSettings(select =>
select.CellSelectionMode(Syncfusion.EJ2.Grids.CellSelectionMode.Box).Mode(Syncfusion.EJ2.Grids.SelectionMode.Cell).Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).Render()
```

```
pe.Multiple)).EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Batch); }).Toolbar(new List<string>() { "Add",
"Update", "Cancel" }).Render()
```

PASTE.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Note: To perform paste functionality, it requires the selection [Mode](#) to be **Cell**, [CellSelectionMode](#) to be **Box** and also Batch Editing should be enabled.

Limitations of Paste Functionality

- Since the string values are not parsed to number and date type, so when the copied string type cells are pasted to number type cells then it will display as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will display as an **empty cell**.

Context menu in ASP.NET MVC Grid Component

The Grid has options to show the context menu when right clicked on it. To enable this feature, you need to define either default or custom item in the [ContextMenuItems](#).

The default items are in the following table.

Items	Description
----	----
AutoFit	Auto fit the current column.
AutoFitAll	Auto fit all columns.
Edit	Edit the current record.
Delete	Delete the current record.
Save	Save the edited record.
Cancel	Cancel the edited state.
Copy	Copy the selected records.
PdfExport	Export the grid data as Pdf document.
ExcelExport	Export the grid data as Excel document.
CsvExport	Export the grid data as CSV document.
Group	Group the current column.
Ungroup	Ungroup the current column.

|SortAscending | Sort the current column in ascending order.|

|SortDescending | Sort the current column in descending order.|

|FirstPage | Go to the first page.|

|PrevPage | Go to the previous page.|

|LastPage | Go to the last page.|

|NextPage | Go to the next page.|

CSHTML

```
@Html.EJS().Grid("ContextMenu").DataSource((IEnumerable<object>)ViewBag.Data
Source).AllowSorting(true).AllowExcelExport(true).AllowPdfExport(true).EditS
ettings(edit =>
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true)).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlig
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().ContextMenuItems(new List<object>() { "AutoFit",
"AutoFitAll", "SortAscending", "SortDescending", "Copy", "Edit", "Delete",
"Save", "Cancel", "PdfExport", "ExcelExport", "CsvExport", "FirstPage",
"PrevPage", "LastPage", "NextPage" }).Render()
```

CONTEXTMENU.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Custom context menu items

The custom context menu items can be added by defining the [ContextMenuItems](#) as a collection of **ContextMenuitemModel**. Actions for this customized items can be defined in the [ContextMenuClick](#) event.

CSHTML

```
@{
    List<object> ContextMenuItems = new List<object>();
}
```

```

        ContextMenuItems.Add(new { text = "Copy with headers", target = ".e-
content", id = "copywithheader" });
    }
    @Html.EJS().Grid("Customcontext").DataSource((IEnumerable<object>)ViewBag.Data
taSource).Columns(col =>
    {
        col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
        col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

        col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
        col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
    }).AllowPaging().ContextMenuItems(ContextMenuItems).ContextMenuClick("Contex
tMenuClick").Render()
<script>
    function ContextMenuClick(args) {
        if (args.item.id === 'copywithheader') {
            this.copy(true);
        }
    }
</script>

```

CUSTOMCONTEXTMENU.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Note: You can hide or show an item in context menu for specific area inside of grid by defining the **Target** property.

Show context menu on left click

By default, the context menu items will be shown in the Grid using the right mouse click action. Show the context menu items during the left mouse click action using the [Created](#) and context menu's [beforeOpen](#) events of the Grid.

Using the `onclick` eventlistener of Grid, you can get the clicked position values and send them to the `open` method of the context menu in the `onclick` event of the Grid. Also, we have prevented the default right click action to open the context menu items using the `Created` event of the Grid.

This is demonstrated in the following sample.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.dataSource)
.Created("created").EditSettings(edit =>
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true)).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).AllowPaging().ContextMenuItems(new List<object>() { "Copy", "Edit",
"Delete", "Save"}).Render()
<script>
function created() {
    var grid = document.getElementById("Grid").ej2_instances[0];
    grid.contextMenuModule.contextMenu.beforeOpen = (args) => {
        if (args.event && args.event.which === 3) args.cancel = true;
        args.event = values;
        grid.contextMenuModule.contextMenuBeforeOpen(args);
    };
}
document.getElementById('Grid').onclick = (event) => {
    values = event;
    var grid = document.getElementById("Grid").ej2_instances[0];
    grid.contextMenuModule.contextMenu.open(
        values.pageY + pageYOffset,
        values.pageX + pageXOffset
    );
};
</script>
```

CONTEXTMENUCLICK.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Enable or disable context menu items

It is possible to enable or disable the default and custom context menu items in the Grid component. This is achieved by using the `enableItems` method of the ContextMenu. To enable or disable menu items, set the `enable` parameter in the `enableItems` method to true, and vice versa.

In the following sample, the Copy item is enabled or disabled based on some condition (as per the needs of the application) in the [RowSelected](#) event of the Grid.

CSHTML


```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.RowSelected("rowSelected").EditSettings(edit =>
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true)).Columns(col
=>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShippedDate").HeaderText("Shipped
Date").Width("140").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().ContextMenuItems(new List<object>() { "Copy", "Edit",
"Delete"}).Render()
<script>
function rowSelected(args) {
    var grid = document.getElementById("Grid").ej2_instances[0];
    var contextMenuObj = grid.contextMenuModule.contextMenu;
    if (args.data.OrderID % 2 === 0) {
        contextMenuObj.enableItems(['Copy'], false);
    } else {
        contextMenuObj.enableItems(['Copy'], true);
    }
}
</script>
```

CONTEXTMENU.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Loading Animation ASP.NET MVC Grid Component

The grid has an option to show a loading indicator in-between the time of fetching the data and binding it to the grid during initial rendering or refreshing or after performing any grid action like sorting, filtering, grouping, and more. The grid supports two indicator types, which is achieved by setting the `loadingIndicator.indicatorType` property to `Spinner` or `Shimmer`. The default value of the indicator type is "Spinner."

In the following sample, the Shimmer indicator is displayed while the grid is loading and refreshing when using the remote data.

CSHTML

```
@Html.EJS().Grid("Grid").AllowSorting(true).AllowFiltering(true).DataSource(
    dataManger =>
    {
        dataManger.Url("https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders/").Adaptor("ODataAdaptor");
        }).Columns(col =>
        {
            col.Field("OrderID").HeaderText("Order ID").Width("130").Add();
            col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();
            col.Field("EmployeeID").HeaderText("Employee ID").Width("115").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
            col.Field("Freight").HeaderText("Freight").Width("150").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
            col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
        })
        .AllowPaging().PageSettings(page=>page.PageCount(3)).LoadingIndicator(l =>
        { l.IndicatorType(Syncfusion.EJ2.Grids.IndicatorType.Shimmer); }).Render()
```

LOADING-ANIMATION.CS

```
public IActionResult Index()
{
    return view();
}
```

Style and Appearance

To modify the Grid appearance, you need to override the default CSS of grid. Find the CSS structure that can be used to modify the Grid appearance. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Customizing the Grid root element

Use the below CSS to customize the Grid root element.

```
`css
.e-grid {
font-family: cursive;
}
`
```

You can modify the grid styling appearance by overriding the default CSS style of the Grid.

In the following sample, the font family of grid content is changed to **cursive**, and the background color of rows, selected rows, alternate rows, and row hovering color is modified using the below CSS styles.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>) ViewBag.DataSource)
.Columns(col =>
{
```

```

        col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("100").Add();
        col.Field("CustomerID").HeaderText("Customer ID").Width("120").Add();

col.Field("Freight").HeaderText("Freight").Format("C2").TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Width("100").Add();
        col.Field("ShipName").HeaderText("Ship Name").Width("180").Add();
    }).AllowPaging().PageSettings(page => { page.PageSize(8);
    }).SelectionSettings(select =>
select.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).Render()
<style>
    .e-grid {
        font-family: cursive;
    }
    .e-grid .e-row:hover .e-rowcell {
        background-color: rgb(204, 229, 255) !important;
    }
    .e-grid .e-rowcell.e-selectionbackground {
        background-color: rgb(230, 230, 250);
    }
    .e-grid .e-row.e-altrow {
        background-color: rgb(150, 212, 212);
    }
    .e-grid .e-row {
        background-color: rgb(180, 180, 180);
    }
</style>

```

STYLE-AND-APPEARANCE.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}

```

How To

Migration from Essential JS 1

This article describes the API migration process of Grid component from Essential JS 1 to Essential JS 2.

Sorting

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| Default | **Property:** *AllowSorting*

@(Html.EJ().Grid<object>("Grid")
 .AllowSorting().Columns(col
=>{
 col.Field("OrderID").Add();
})| **Property:** *AllowSorting*

@Html.EJS().Grid("Grid")
 .AllowSorting().Columns(col =>{
 col.Field("OrderID").Add();
}).Render()|

| Clear the Sorted columns | **Method:** *clearSorting()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data

Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.clearSorting();
</script>| **Method:**
clearSorting()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.clearSorting()
</script>
| Get the Sorted Columns by using the Fieldname | **Method:** *getsortColumnByField(field)*

@(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getsortColumnByField("OrderID");
</script>| **Property:**
sortSettings.columns

Sorted Column for a particular field can be get by iterating the
sortSettings.columns with the fieldname

| Remove the Sorted Columns | **Method:** *removeSortedColumns(fieldName)*

@(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.removeSortedColumns("OrderID");
</script>| **Method:**
removeSortColumn(fieldName)

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<
object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document
.getElementById('Grid').ej2_instances[0];
gridObj.removeSortColumn("OrderI
D")
</script>

| Sort a Column by using the method | **Method:** *sortColumn(columnName, [sortingDirection])*

@(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.sortColumn("OrderID", "ascending");
</script>|
Method: *sortColumn(columnName,
Direction)*

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBa
g.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.sortColumn("OrderID",
"ascending");
</script>

Grouping

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowGrouping*

@(Html.EJ().Grid<object>("Grid")
 .AllowGrouping().Columns(col =>{
 col.Field("OrderID").Add();
})
)|

Property:

AllowGrouping

@Html.EJS().Grid("Grid")
 .AllowGrouping().Columns(col
=>{
 col.Field("OrderID").Add();
}).Render()|

| Group Columns initially | **Property:** *AllowGrouping.GroupedColumns*

@(Html.EJ().Grid<object>("Grid")
 .AllowGrouping()
.GroupSettings(group => {
group
.GroupedColumns(col => { col.Add("ShipCountry"); }); })
.Columns(col
=>{
 col.Field("OrderID").Add();
 col.Field("ShipCountry").Add();
 col.Field("ShipCity").Add();
})
)| **Property:**
AllowGrouping.Columns

@Html.EJS().Grid("Grid")
.AllowGrouping()
.GroupSettin

```
gs(group=><br>group.Columns(new string[] { "ShipCountry" }));<br>.Columns(col =>{<br>
    &#160;&#160;col.Field("OrderID").Add();<br> &#160;&#160;col.Field("ShipCountry").Add();<br>
    &#160;&#160;col.Field("ShipCity").Add();<br>}).Render();
```

| Caption Template | **Property:** *GroupSettings.CaptionFormat*

 @(Html.EJ().Grid<object>("Grid")
.AllowGrouping())
.GroupSettings(group
 =>{
group.CaptionFormat("#template"); })
Script:
<script id="template"
 type="text/x-jsrender">
You can add template elements here
</script>| **Property:**
GroupSettings.CaptionTemplate

@Html.EJS().Grid("Grid")
 .AllowGrouping().Gr
 oupSettings(
group =>{group.Columns(
new string[] { "Freight" }

 .CaptionTemplate("#captiontemplate");
 })
.Render()
Script:
<script>
<script id="captiontemplate"
type="text/x-
 template">
You can add template elements here
</script>

| Show Drop Area | **Property:** *AllowGrouping.ShowDropArea*

 @(Html.EJ().Grid<object>("Grid")
.AllowGrouping())
.GroupSettings(group
 =>{
group.ShowDropArea(false) }); }
.Columns(col
 =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
})
)|
Property:
AllowGrouping.showDropArea

@Html.EJS().Grid("Grid")
 .AllowGrouping()
.
 GroupSettings(group=>
group.ShowDropArea(false))
.Columns(col =>{

 col.Field("OrderID").Add();

 col.Field("ShipCity").Add();
}).Render();

| Collapse all group caption rows | **Method:** *collapseAll()*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.collapseAll();
</script>| **Method:**
collapseAll()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
View
 Bag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.groupModule.collapseAll()<
 br></script>

| Expand all group caption rows | **Method:** *expandAll()*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.expandAll();
</script>| **Method:**
expandAll()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
View
 Bag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.groupModule.expandAll()<b
 r></script>

| Expand or collapse the row based on the row state in grid | **Method:**
expandCollapse(\$target)

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.expandCollapse(\$("#tr td.recordplusexpand >

div").first());
</script> | **Method:**
`expandCollapseRows()`
</code>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.groupModule.expandCollap
seRows(
gridObj.getContent().querySelectorAll
('e-recordplusexpand')[0])
</script>
</code>
 | Collapse the group drop area in grid | **Method:** `collapseGroupDropArea()`
</code>
 @Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.collapseGroupDropArea();
</script> | Not Applicable
 | Expand the group drop area in grid | **Method:** `expandGroupDropArea()`
</code>
 @Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.expandGroupDropArea();
</script> | Not Applicable
 | Group a column by using the method | **Method:** `groupColumn(fieldName)`
</code>
 @Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.groupColumn("OrderID");
</script> | **Method:**
`groupColumn(fieldName)`
</code>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<objec
t>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.groupColumn("OrderID")
</script>
</code>
 | Ungroup a grouped column by using the method | **Method:** `ungroupColumn(fieldName)`
</code>
 @Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.ungroupColumn("OrderID");
</script> | **Method:**
`ungroupColumn(fieldName)`
</code>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<obj
ect>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.ungroupColumn("OrderID")
</script>
</code>

Filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** `AllowFiltering`
</code>
 @Html.EJ().Grid<object>("Grid")
.AllowFiltering().Columns(col
=>{
 col.Field("OrderID").Add();
})
 | **Property:**
`AllowFiltering`
</code>@Html.EJS().Grid("Grid")
 .AllowFiltering().Columns(col =>{
 col.Field("OrderID").Add();
}).Render()
 | Menu Filtering | **Property:** `FilterSettings.FilterType`
</code>
 @Html.EJ().Grid<object>("Grid")
.AllowFiltering()
.FilterSettings(filter
=>{
filter.FilterType(FilterType.Menu);
})
.Columns(col
=>{
 col.Field("OrderID").Add();
})
 | **Property:**

```
FilterSettings.Type<br/><br/>@Html.EJS().Grid("Grid")<br>&#160;.AllowFiltering().FilterSettings(fil
ter=><br>filter.Type(Syncfusion.EJ2.Grids<br>.FilterType.Menu))<br>.Columns(col =><br>
&#160;&#160;col.Field("OrderID").Add();<br> &#160;&#160;col.Field("ShipCountry").Add();<br>
&#160;&#160;col.Field("ShipCity").Add();<br>)).Render()
```

| Excel Filtering | **Property:** *FilterSettings.FilterType*


```
@(Html.EJ().Grid<object>("Grid")<br>.AllowFiltering()<br>.FilterSettings(filter =>
{<br>filter.FilterType(FilterType.Excel); }<br>.Columns(col
=><br>&#160;col.Field("OrderID").Add();<br>)) | Property:
```

```
FilterSettings.Type<br/><br/>@Html.EJS().Grid("Grid")<br>&#160;.AllowFiltering().FilterSettings(fil
ter=><br>filter.Type(Syncfusion.EJ2.Grids<br>.FilterType.Excel))<br>.Columns(col =><br>
&#160;&#160;col.Field("OrderID").Add();<br> &#160;&#160;col.Field("ShipCountry").Add();<br>
&#160;&#160;col.Field("ShipCity").Add();<br>)).Render()
```

| Clear the Filtered values | **Method:** *clearFiltering(field)* - *field is optional*


```
@(Html.EJ().Grid<object>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.clearFiltering();<br></script> | Method:
clearFiltering()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>Vi
ewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.clearFiltering()<br></script>
```

| Filter a column by using the method | **Method:** *filterColumn(fieldName, filterOperator, filterValue, predicate, [matchcase],[actualFilterValue])*


```
@(Html.EJ().Grid<object>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.filterColumn("OrderID",<br>"equal","10248","and",
true);<br></script> | Method: filterByColumn(fieldName, <br>filterOperator, filterValue, predicate,
matchCase,<br>ignoreAccent, actualFilterValue,
actualOperator)<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>V
iewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.filterByColumn("OrderID",<br>
equal,10248)<br></script>
```

| Filter columns by Collection | **Method:** *filterColumn(filterCollection)*


```
@(Html.EJ().Grid<object>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.filterColumn([{field:"OrderID",<br>operator:"lessthan",val
ue:"10266",<br>predicate:"and",matchcase:true},{field:"EmployeeID",operator:<br>"equal"
,value:2,predicate:"and",matchcase:true}])<br></script> | Property:
filterSettings.columns<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)
<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.filterSettings = {columns: [{
field: 'ShipCity', matchCase: false, operator: 'startswith', predicate: 'and', value: 'reims'
}]}<br></script>
```


| Get the Filtered Records | **Method:** *getFilteredRecords()*

```
@(Html.EJ().Grid<object>(<br>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br><b>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getFilteredRecords();<br></script>| Not Applicable
```

Searching

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *ToolBarSettings.ToolBarItem*s

 @(<Html.EJ().Grid<object>("Grid")

 .ToolBarSettings(toolBar =>
toolBar.ShowToolBar()
.ToolBarItems(items
=>
{items.AddTool(ToolBarItems.Search);})
.Columns(col
=>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
})
)|
Property: *ToolBar*

@Html.EJS().Grid("Grid")
 .ToolBar(new List<string>() {
"Search" })
.Columns(col =>{
 col.Field("OrderID").Add();

 col.Field("ShipCity").Add();
}).Render()

| Clear the Searched values | **Method:** *clearSearching()*

```
@(Html.EJ().Grid<object>(<br>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br><b>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.clearSearching();<br></script>| Method:
searchModule.search()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>
)<br>ViewBag.dataSource)<br>.Render()<b>Script:<br><script><br>var gridObj =
document.getElementById(<br>('Grid')).ej2_instances[0];<br>gridObj.searchModule.search("");<b
r></script>
```

| Search a value | **Method:** *search(searchString)*

```
@(Html.EJ().Grid<object>(<br>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br><b>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.search("France");<br></script>| Method:
searchModule.search()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>
)<br>ViewBag.dataSource)<br>.Render()<b>Script:<br><script><br>var gridObj =
document.getElementById(<br>('Grid')).ej2_instances[0];<br>gridObj.searchModule.search("Fran
ce");<br></script>
```

Paging

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowPaging*

 @(<Html.EJ().Grid<object>("Grid")

 .AllowPaging().Columns(col =>{
 col.Field("OrderID").Add();
})
)|
Property: *AllowPaging*

@Html.EJS().Grid("Grid")
 .AllowPaging().Columns(col
=>{
 col.Field("OrderID").Add();
}).Render()

| Customize Paging | **Property:** *PageSettings.PageSize*


```
@(Html.EJ().Grid<object>("Grid")<br> &#160;.AllowPaging()<br>.PageSettings(page=>{
page.PageSize(8).PageSizeList(new List<int>() {5, 10 }));<br>.Columns(col
=>{<br>&#160;col.Field("OrderID").Add();<br>&#160;col.Field("ShipCity").Add();<br>})<br>)|
```


Property:

```

PageSettings.PageSize<br/><br/>@Html.EJS().Grid("Grid")<br/>&#160;.AllowPaging().PageSettings(
page<br/>=> page.PageSize(8).PageSizes(new string[] { "5", "10" })<br/>.Columns(col =>{<br/>
&#160;&#160;col.Field("OrderID").Add();<br/>
&#160;&#160;col.Field("ShipCity").Add();<br/>}).Render()

```

| Change Page Size | **Method:** *changePageSize(pageSize)*


```

@(Html.EJ().Grid<object><br/>("Grid").Datasource<br/>((IEnumerable<object>)<br/>ViewBag.data
Source))<br/>Script:<br/><script><br/>var gridObj =
$("#Grid").data("ejGrid");<br/>gridObj.changePageSize(7);<br/></script>| Property:
pageSettings.pageSize<br/><br/>Pagesize can be modified dynamically by using the below
code<br/>@Html.EJS().Grid("Grid")<br/>.DataSource((IEnumerable<object>)<br/>ViewBag.dataSo
urce)<br/>.Render()<br/>Script:<br/><script><br/>PageSettings.PageSize = 7;<br/></script>

```

| Get Current Page Index | **Method:** *getCurrentIndex()*


```

@(Html.EJ().Grid<object><br/>("Grid").Datasource<br/>((IEnumerable<object>)<br/>ViewBag.data
Source))<br/>Script:<br/><script><br/>var gridObj =
$("#Grid").data("ejGrid");<br/>gridObj.getCurrentIndex();<br/></script>| Property:
pageSettings.currentPage<br/><br/>@Html.EJS().Grid("Grid")<br/>.DataSource((IEnumerable<obje
ct>)<br/>ViewBag.dataSource)<br/>.Render()<br/>Script:<br/><script><br/>var gridObj =
document.getElementById<br/>('Grid').ej2_instances[0];<br/>var currentPage =
gridObj.pageSettings.currentPage;<br/></script>

```

| Get Pager Element | **Method:** *getPager()*


```

@(Html.EJ().Grid<object><br/>("Grid").Datasource<br/>((IEnumerable<object>)<br/>ViewBag.data
Source))<br/>Script:<br/><script><br/>var gridObj =
$("#Grid").data("ejGrid");<br/>gridObj.getPager();<br/></script>| Method:
getPager()<br/><br/>@Html.EJS().Grid("Grid")<br/>.DataSource((IEnumerable<object>)<br/>ViewB
ag.dataSource)<br/>.Render()<br/>Script:<br/><script><br/>var gridObj =
document.getElementById<br/>('Grid').ej2_instances[0];<br/>gridObj.getPager();<br/></script>

```

| Send a paging request to specified Page | **Method:** *gotoPage(pageIndex)*


```

@(Html.EJ().Grid<object><br/>("Grid").Datasource<br/>((IEnumerable<object>)<br/>ViewBag.data
Source))<br/>Script:<br/><script><br/>var gridObj =
$("#Grid").data("ejGrid");<br/>gridObj.gotoPage(3);<br/></script>| Method:
gotoPage(pageIndex)<br/><br/>@Html.EJS().Grid("Grid")<br/>.DataSource((IEnumerable<object>)
<br/>ViewBag.dataSource)<br/>.Render()<br/>Script:<br/><script><br/>var gridObj =
document.getElementById<br/>('Grid').ej2_instances[0];<br/>gridObj.gotoPage(3);<br/></script>

```

| Calculate Pagesize of grid by using its Parent height(containerHeight) | **Method:**

```

calculatePageSizeBy<br/>ParentHeight(containerHeight)<br/><br/>
@(Html.EJ().Grid<object><br/>("Grid").Datasource<br/>((IEnumerable<object>)<br/>ViewBag.data
Source))<br/>Script:<br/><script><br/>var gridObj =
$("#Grid").data("ejGrid");<br/>gridObj.<br/>calculatePageSizeByParentHeight(400);<br/></script>|
Not Applicable

```

Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowSelection*

 @(Html.EJ().Grid<object>("Grid")
 .AllowSelection().Columns(col =>{
 col.Field("OrderID").Add();
})
)|

Property:

AllowSelection

@Html.EJS().Grid("Grid")
 .AllowSelection().Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Single Selection | **Property:** *SelectionType*

 @(Html.EJ().Grid<object>("Grid")
 .AllowSelection()
.SelectionType(SelectionType.Single)
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
})
)|

Property:

SelectionSettings.Type

@Html.EJS().Grid("Grid")
 .AllowSelection()
.Selecti onSettings(select =>
select.Type(Syncfusion.EJ2
.Grids.SelectionType.Single))
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
}).Render()

| Multiple Selection | **Property:** *SelectionType*

 @(Html.EJ().Grid<object>("Grid")
 .AllowSelection()
.SelectionType(SelectionType.Multiple)
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
})
)|

Property:

SelectionSettings.Type

@Html.EJS().Grid("Grid")
 .AllowSelection()
.Selecti onSettings(select =>
select.Type(Syncfusion.EJ2
.Grids.SelectionType.Multiple))
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
}).Render()

| Row Selection | **Property:** *SelectionSettings.SelectionMode*

 @(Html.EJ().Grid<object>("Grid")
 .AllowSelection()
.SelectionSettings(select => {
select.SelectionMode(mode => {
mode.AddMode(SelectionMode.Row); });
})
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
})
)|

Property:

SelectionSettings.Mode

@Html.EJS().Grid("Grid")
 .AllowSelection()
.Selecti onSettings(select =>
select.Mode(Syncfusion.EJ2
.Grids.SelectionMode.Row))
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
}).Render()

| Cell Selection | **Property:** *SelectionSettings.SelectionMode*

 @(Html.EJ().Grid<object>("Grid")
 .AllowSelection()
.SelectionSettings(select => {
select.SelectionMode(mode => {
mode.AddMode(SelectionMode.Cell); });
})
.Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("ShipCity").Add();
})
)| **Property:**

SelectionSettings.Mode

@Html.EJS().Grid("Grid")
 .AllowSelection()
.Selecti onSettings(select

```
=><br>select.Mode(Syncfusion.EJ2<br>.Grids.SelectionMode.Cell))<br>.Columns(col =>{<br>
&#160;&#160;col.Field("OrderID").Add();<br>
&#160;&#160;col.Field("ShipCity").Add();<br>}).Render()
```

| Clear the selected Cells | **Method:** *clearCellSelection()*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.clearCellSelection();<br></script> | Method:
clearCellSelection()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object><br>
ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.selectionModule<br>.clearC
ellSelection()<br></script>
```

| Clear the selected Columns | **Method:** *clearColumnSelection([index])*-
index is optional


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.clearColumnSelection();<br></script> | Not Applicable
```

| Get the selected Records | **Method:** *getSelectedRecords()*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getSelectedRecords();<br></script> | Method:
getSelectedRecords()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object><br>
ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document<br>.getElementById('Grid')<br>.ej2_instances[0]<br>gridObj.getSelectedRecords();<br>
</script>
```

| Get the selected Rows | **Method:** *getSelectedRows()*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getSelectedRows();<br></script> | Method:
getSelectedRows()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object><br>
ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document<br>.getElementById('Grid')<br>.ej2_instances[0]<br>gridObj.getSelectedRows();<br>
</script>
```

| Select Cells | **Method:** *selectCells(rowCellIndexes)*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.selectCells([[1, [4, 3, 2]]]);<br></script> | Method:
selectionModule.selectCells();<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<o
bject><br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.selectionModule.selectCells
<br>([[{ rowIndex: 0, cellIndexes: [0] },<br>{ rowIndex: 1, cellIndexes: [1] }]);<br></script>
```

| Select Rows | **Method:** *selectRows(fromIndex, toIndex)*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
```

`$("#Grid").data("ejGrid");
gridObj.selectRows(1, 4);
</script>` | **Method:**
`selectionModule.selectRows()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.selectionModule.
select
Rows([0, 2])
</script>`

| Triggers when a
cell is selected | **Event:** `CellSelected

`
`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.CellSelected("cellSelected");}))
Script:
<script>
function
cellSelected(){
</script>` | **Event:**
`CellSelected

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.CellSelected("cellSelected")
.Render()
Script:
<script>
functi
on cellSelected(){
</script>`

| Triggers before the cell is being selected | **Event:** `CellSelecting

`
`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.CellSelecting("cellSelecting");}))
Script:
<script>
function
cellSelecting(){
</script>` | **Event:**
`CellSelecting

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.CellSelecting("cellSelecting")
.Render()
Script:
<script>
functi
on cellSelecting(){
</script>`

| Triggers when a
cell is deselected | **Event:** `CellDeselected

`
`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.CellDeselected("cellDeselected");}))
Script:
<script>
function
cellDeselected(){
</script>` | **Event:**
`CellDeselected

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.CellDeselected("cellDeselected")
.Render()
Script:
<script>
function cellDeselected(){
</script>`

| Triggers before the cell is being deselected | **Event:** `CellDeselecting

`
`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.CellDeselecting("cellDeselecting");}))
Script:
<script>
function
cellDeselecting(){
</script>` | **Event:**
`CellDeselecting

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.CellDeselecting("cellDeselecting")
.Render()
Script:
<script>
function cellDeselecting(){
</script>`

| Triggers when the
row is selected | **Event:** `RowSelected

`
`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.RowSelected("rowSelected");}))
Script:
<script>
function
rowSelected(){
</script>` | **Event:**

```
RowSelected<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.RowSelected("rowSelected")<br>.Render()<br><b>Script:<br><script><br>function rowSelected(){<br></script></b>
```

```
| Triggers before the row is being selected | Event: RowSelecting<br/><br/>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
=>{<br>eve.RowSelecting("rowSelecting");}))<br><b>Script:<br><script><br>function
rowSelecting(){<br></script> | Event:
RowSelecting<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.RowSelecting("rowSelecting")<br>.Render()<br><b>Script:<br><script><br>function
rowSelecting(){<br></script>
```

```
| Triggers when the <br>row is deselected | Event: RowDeselected<br/><br/>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
=>{<br>eve.RowDeselected("rowDeselected");}))<br><b>Script:<br><script><br>function
rowDeselected(){<br></script> | Event:
RowDeselected<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.RowDeselected("rowDeselected")<br>.Render()<br><b>Script:<br><script><br>function
rowDeselected(){<br></script>
```

```
| Triggers before the row is being deselected | Event: RowDeselecting<br/><br/>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
=>{<br>eve.RowDeselecting("rowDeselecting");}))<br><b>Script:<br><script><br>function
rowDeselecting(){<br></script> | Event:
RowDeselecting<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.RowDeselecting("rowDeselecting")<br>.Render()<br><b>Script:<br><script><br>function
rowDeselecting(){<br></script>
```

```
| Triggers when the column is selected | Event: ColumnSelected<br/><br/>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
=>{<br>eve.ColumnSelected("columnSelected");}))<br><b>Script:<br><script><br>function
columnSelected(){<br></script> | Not Applicable
```

```
| Triggers before the column is being selected | Event: ColumnSelecting<br/><br/>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
=>{<br>eve.ColumnSelecting("columnSelecting");}))<br><b>Script:<br><script><br>function
columnSelecting(){<br></script> | Not Applicable
```

```
| Triggers when the column is deselected | Event: ColumnDeselected<br/><br/>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
=>{<br>eve.ColumnDeselected("columnDeselected");}))<br><b>Script:<br><script><br>function
columnDeselected(){<br></script> | Not Applicable
```

| Triggers before the column is being deselected | **Event:** *ColumnDeselecting*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.ColumnDeselecting("columnDeselecting");}))
Script:
<script>
function
 columnDeselecting(){
</script>| Not Applicable

Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *EditSettings*

 @(Html.EJ().Grid<object>("Grid")

 .EditSettings(edit => { edit.AllowAdding()
.AllowDeleting().AllowEditing();
 })
 col.Field("OrderID").Add();
})
| **Property:**
EditSettings

@Html.EJS().Grid("Grid")
 .EditSettings(edit => {
 edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true) })
.Columns(col =>{

 col.Field("OrderID").Add();
}).Render()|

| Inline Editing | **Property:** *EditSettings.EditMode*

 @(Html.EJ().Grid<object>("Grid")

 .EditSettings(edit => {
 edit.AllowAdding()
.AllowDeleting().AllowEditing()
.EditMode(EditMode.Normal);
 })
 col.Field("OrderID").Add();
})
| **Property:**
EditSettings.Mode

@Html.EJS().Grid("Grid")
 .EditSettings(edit => {
 edit.AllowAdding(true).
AllowEditing(true).AllowDeleting(true).Mode
{Syncfusion.EJ2.Gri
 ds.EditMode.Normal); })
.Columns(col =>{

 col.Field("OrderID").Add();
}).Render()|

| Dialog Editing | **Property:** *EditSettings.EditMode*

 @(Html.EJ().Grid<object>("Grid")

 .EditSettings(edit => {
 edit.AllowAdding()
.AllowDeleting().AllowEditing()
.EditMode(EditMode.Dialog);
 })
 col.Field("OrderID").Add();
})
| **Property:**
EditSettings.Mode

@Html.EJS().Grid("Grid")
 .EditSettings(edit => {
 edit.AllowAdding(true).
AllowEditing(true).AllowDeleting(true).Mode
{Syncfusion.EJ2.Gri
 ds.EditMode.Dialog); })
.Columns(col =>{

 col.Field("OrderID").Add();
}).Render()|

| Batch Editing | **Property:** *EditSettings.EditMode*

 @(Html.EJ().Grid<object>("Grid")

 .EditSettings(edit => {
 edit.AllowAdding()
.AllowDeleting().AllowEditing()
.EditMode(EditMode.Batch);
 })
 col.Field("OrderID").Add();
})
| **Property:**
EditSettings.Mode

@Html.EJS().Grid("Grid")
 .EditSettings(edit => {
 edit.AllowAdding(true).
AllowEditing(true).AllowDeleting(true).Mode
{Syncfusion.EJ2.Gri
 ds.EditMode.Batch); })
.Columns(col =>{

 col.Field("OrderID").Add();
}).Render()|

| Add a new Record | **Method:** *addRecord([data],[serverChange])*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.dataSource)
Script:
<script>
var gridObj =

`$("#Grid").data("ejGrid");
gridObj.addRecord({"OrderID":12333});
</script>` | **Method:**
addRecord(data(optional),
*index(optional))
</br>*@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.addRecord();
</script>

| Batch Cancel | **Method:** *batchCancel()*
</br>
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.batchCancel();
</script> | Not Applicable

| Batch Save | **Method:** *batchSave()*
</br>
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.batchSave();
</script> | **Method:**
batchSave()
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.editModule.batchSave()
</script>

| Save a Cell - If *preventSaveEvent* is true, then it
will prevent the client side cellSave event |
Method: *saveCell([preventSaveEvent])
</br>*
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.saveCell();
</script> | **Method:**
saveCell()
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.editModule.saveCell()
</script>

| End Edit | **Method:** *endEdit()*
</br>
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.endEdit();
</script> | **Method:**
endEdit()
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.endEdit()
</script>

| Cancel Edit | **Method:** *cancelEdit()*
</br>
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.cancelEdit();
</script> | **Method:**
closeEdit()
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.closeEdit()
</script>

| Delete Record | **Method:** *deleteRecord(fieldName, data)
</br>*
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))


```

Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.deleteRecord("OrderID",<br>{ OrderID: 10249,
EmployeeID: 3 });<br></script>| Method: deleteRecord(field, data)- Field and<br> data are
optional<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag
.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.deleteRecord()<br></script>
| Delete Row | Method: deleteRow($tr)<br><br>
@Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.deleteRow($("#.gridcontent tr").first());<br></script>|
Method:
deleteRow(tr)<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>View
Bag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.deleteRow(grid.getContentTable()<br>.querySelector("tbody").firstChild)<br></script>
| Edit a cell in Batch edit mode | Method: editCell(index, fieldName)<br><br>
@Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.editCell(2, "OrderID");<br></script>| Method:
editModule.editCell(index,
field)<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.data
Source)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.editModule.editCell(0,gridObj.columns[0].field)<br></script>
| Edit Form Validation | Method: editFormValidate()<br><br>
@Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.editFormValidate();<br></script>| Method:
editModule.formObj.validate()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<
object>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.editModule.formObj.validate()<br></script>
| Get Batch Changes | Method: getBatchChanges()<br><br>
@Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getBatchChanges();<br></script>| Method:
editModule.getBatchChanges()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<
object>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.editModule.getBatchChanges()<br></script>
| Refresh Batch Edit Changes | Method: refreshBatchEditChanges()<br><br>
@Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.data

```



```

Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.refreshBatchEditChanges();<br></script> | Not Applicable

| Set Default Data for adding | Method: setDefaultData(defaultData)<br><br>
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj = $("#Grid").data("ejGrid");<br>var defaultData =
{OrderID:"10000"};<br>gridObj.setDefaultData(defaultData);<br></script> | Method:
editModule.addRecord()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>
)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.editModule.addRecord({Or
derID:10000})<br></script>

| Start Edit | Method: startEdit($tr)<br><br>
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.startEdit($("#.gridcontent tr")<br>.first());<br></script> |
Method:
startEdit()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object><br>ViewB
ag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.startEdit(gridObj.selectRow
(0))<br></script>

| Update Record | Method: updateRecord(fieldName, data)<br><br>
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.updateRecord("OrderID",<br>{ OrderID: 10249,
EmployeeID: 3 });<br></script> | Not Applicable

| Set Cell value | Method: setCellText()<br><br>
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.setCellText(0, 1, "France");<br></script> | Method:
setCellValue(key, field,
value)<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object><br>ViewBag.d
ataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.setCellValue(10248,"Custo
merID","A")<br></script>

| Get Currently edited cell value | Method: getCurrentEditCellData()<br><br>
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getCurrentEditCellData();<br></script> | Method:
getCurrentEditCellData()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>
)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.<br>editModule.getCurrent
EditCellData();<br></script>

```

| Triggers when adding a
record in batch editing | **Event:** *BatchAdd*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.BatchAdd("BatchAdd");}))
Script:
<script>
function
 batchAdd(){
</script>| **Event:**
BatchAdd

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.BatchAdd("batchAdd")
.Render()
Script:
<script>
function
 batchAdd(){
</script>

| Triggers when deleting a
record in batch editing | **Event:** *BatchDelete*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.BatchDelete("batchDelete");}))
Script:
<script>
function
 batchDelete(){
</script>| **Event:**
BatchDelete

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.BatchDelete("batchDelete")
.Render()
Script:
<script>
function
 batchDelete(){
</script>

| Triggers before adding a record in batch editing | **Event:** *BeforeBatchAdd*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.BeforeBatchAdd("beforeBatchAdd");}))
Script:
<script>
function
 beforeBatchAdd(){
</script>| **Event:**
BeforeBatchAdd

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.BeforeBatchAdd("beforeBatchAdd")
.Render()
Script:
<script>
 >
function beforeBatchAdd(){
</script>

| Triggers before deleting a record in batch editing | **Event:** *BeforeBatchDelete*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.BeforeBatchDelete("beforeBatchDelete");}))
Script:
<script>
function
 beforeBatchDelete(){
</script>| **Event:**
BeforeBatchDelete

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.BeforeBatchDelete("beforeBatchDelete")
.Render()
Script:

 ><script>
function beforeBatchDelete(){
</script>

| Triggers before saving a record in batch editing | **Event:** *BeforeBatchSave*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.BeforeBatchSave("beforeBatchSave");}))
Script:
<script>
function
 beforeBatchSave(){
</script>| **Event:**
BeforeBatchSave

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.BeforeBatchSave("beforeBatchSave")
.Render()
Script:
<script>
 >
function beforeBatchSave(){
</script>

| Triggers before the
record in being edited | **Event:** *BeginEdit*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)

```

ce)<br>.ClientSideEvents(eve
=>{<br>eve.BeginEdit("beginEdit");}))<br>Script:<br><script><br>function
beginEdit(){<br></script>| Event:
BeginEdit<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag
g.dataSource)<br>.BeginEdit("beginEdit")<br>.Render()<br>Script:<br><script><br>function
beginEdit(){<br></script>

|Triggers when the <br>cell is being edited| Event: CellEdit<br><br>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.CellEdit("cellEdit");}))<br>Script:<br><script><br>function cellEdit(){<br></script>|
Event:
CellEdit<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.
dataSource)<br>.CellEdit("cellEdit")<br>.Render()<br>Script:<br><script><br>function
cellEdit(){<br></script>

|Triggers when the <br>cell is saved| Event: CellSave<br><br>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.CellSave("cellSave");}))<br>Script:<br><script><br>function cellSave(){<br></script>|
Event:
CellSave<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag
.dataSource)<br>.cellSave("cellSave")<br>.Render()<br>Script:<br><script><br>function
cellSave(){<br></script>

|Triggers when the record is added| Event: EndAdd<br><br>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.EndAdd("endAdd");}))<br>Script:<br><script><br>function endAdd(){<br></script>|
Event:
ActionComplete<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>V
iewBag.dataSource)<br>.ActionComplete("actionComplete")<br>.Render()<br>Script:<br><script
><br>function actionComplete(){<br></script>

|Triggers when the record is deleted| Event: EndDelete<br><br>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.EndDelete("endDelete");}))<br>Script:<br><script><br>function
endDelete(){<br></script>| Event:
ActionComplete<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>V
iewBag.dataSource)<br>.ActionComplete("actionComplete")<br>.Render()<br>Script:<br><script
><br>function actionComplete(){<br></script>

|Triggers after the record is edited| Event: EndEdit<br><br>
@ (Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.EndEdit("endEdit");}))<br>Script:<br><script><br>function endEdit(){<br></script>|

```

Event:

```
ActionComplete<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ActionComplete("actionComplete")<br>.Render()<br><b>Script:<br><script><br><br>function actionComplete(){<br></script></b>
```

Resizing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowResizing*

 @Html.EJ().Grid<object>("Grid")
 .AllowResizing().Columns(col =>{
 col.Field("OrderID").Add();
})
|

Property:

```
AllowResizing<br/><br/>@Html.EJS().Grid("Grid")<br>&#160;.AllowResizing().Columns(col =>{<br>&#160;&#160;col.Field("OrderID").Add();<br>}).Render()|
```

| Resize a column by using the method | **Method:** *resizeColumns(column,width)*


```
@(Html.EJ().Grid<object>("Grid").DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br><b>Script:<br><script><br>var gridObj =
```

```
$("#Grid").data("ejGrid");<br>gridObj.resizeColumns("OrderID",width);<br></script>| Property: columns.width<br><br>To resize a column, set width to that particular column and then refresh the grid by using the refresh
```

```
method.<br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.Render()<br><b>Script:<br><script><br>var gridObj =  
document.getElementById("Grid").ej2_instances[0];<br>gridObj.columns[1].width =  
100;<br>gridObj.refresh();<br></script>
```

| Triggers when a column resize starts | **Event:** *ResizeStart*


```
@(Html.EJ().Grid<object>("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
```

```
=>{<br>eve.ResizeStart("resizeStart");}))<br><b>Script:<br><script><br>function  
resizeStart(){<br></script>| Event:
```

```
ResizeStart<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ResizeStart("resizeStart")<br>.Render()<br><b>Script:<br><script><br>function  
resizeStart(){<br></script>
```

| Triggers when a column is resized | **Event:** *Resized*


```
@(Html.EJ().Grid<object>("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
```

```
=>{<br>eve.Resized("resized");}))<br><b>Script:<br><script><br>function resized(){<br></script>|
```

Event:

```
ResizeStop<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ResizeStop("resizeStop")<br>.Render()<br><b>Script:<br><script><br>function  
resizeStop(){<br></script>
```

| Triggers when a column resize stops | **Event:** *ResizeEnd*


```
@(Html.EJ().Grid<object>("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.ClientSideEvents(eve
```

```
=>{<br>eve.ResizeEnd("resizeEnd");})<br>Script:<br><script><br>function
resizeEnd(){<br></script>| Not Applicable
```

Reordering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowReordering*

@(Html.EJ().Grid<object>("Grid")
 .AllowReordering().Columns(col =>{
 col.Field("OrderID").Add();
})
)|

Property:

AllowReordering

@Html.EJS().Grid("Grid")
 .AllowReordering().Columns(col =>{
 col.Field("OrderID").Add();
}).Render()|

| Reorder Columns| **Method:** *reorderColumns(fromFieldName, toFieldName)*

@(Html.EJ().Grid<object>("Grid").Datasource
((IEnumerable<object>)
ViewBag.data Source))
Script:
<script>
var gridObj = \$("#Grid").data("ejGrid");
gridObj.reorderColumns("OrderID", "CustomerID");
</script>|

Method: *reorderColumns(fromFieldName, toFieldName)*

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj = document.getElementById("Grid").ej2_instances[0];
gridObj.reorderColumns("OrderID", "CustomerID");
</script>

| Reorder Rows| **Method:** *reorderRows(indexes, toIndex)*

@(Html.EJ().Grid<object>("Grid").Datasource
((IEnumerable<object>)
ViewBag.data Source))
Script:
<script>
var gridObj = \$("#Grid").data("ejGrid");
gridObj.reorderRows([0,1],3);
</script>| Not Applicable

Context Menu

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *ContextMenuSettings.EnableContextMenu*

@(Html.EJ().Grid<object>("Grid")
 .ContextMenuSettings(contextMenu =>{contextMenu.EnableContextMenu();})
.Columns(col

=>{
 col.Field("OrderID").Add();
})
)| **Property:** *ContextMenuItems*

@Html.EJS().Grid("Grid")
 .ContextMenuItems(new List<object>() { "AutoFit", "AutoFitAll" })
.Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Triggers when context menu item is clicked| **Event:** *ContextClick*

@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve =>{
eve.ContextClick("contextClick");})
Script:
<script>
function contextClick(){
</script>| **Event:** *ContextMenuClick*

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)
ViewBag.dataSource)
.ContextMenuClick("contextMenuClick")
.Render()
Script:
<script>
function contextMenuClick(){
</script>

| Triggers when context menu opens | **Event:** *ContextOpen*
 @(Html.EJ().Grid<object>("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.ContextOpen("contextOpen");}))
Script:
<script>
function
 contextOpen(){
</script> | **Event:**
ContextMenuOpen

@(Html.EJ().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.ContextMenuOpen("contextMenuOpen")
.Render())
Script:
<script>
function contextMenuOpen(){
</script>

Toolbar

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Print | **Property:** *ToolbarSettings.ToolbarItems*

 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>{
toolbar.ShowToolbar()
.ToolbarItems(items=>{
items.AddTool(ToolBarItems.Print
 Grid);});
})
.Columns(col =>{
 col.Field("OrderID").Add();
}) | **Property:**
Toolbar

@(Html.EJ().Grid("Grid")
 .Toolbar(new List<string>() { "Print"
 })
.Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Add | **Property:** *ToolbarSettings.ToolbarItems*

 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>{
toolbar.ShowToolbar()
.ToolbarItems(items=>{
items.AddTool(ToolBarItems.Add
);});
})
.Columns(col =>{
 col.Field("OrderID").Add();
}) | **Property:**
Toolbar

@(Html.EJ().Grid("Grid")
 .Toolbar(new List<string>() { "Add"
 })
.Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Edit | **Property:** *ToolbarSettings.ToolbarItems*

 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar =>{
toolbar.ShowToolbar()
.ToolbarItems(items
 =>{
items.AddTool(ToolBarItems.Edit);});
})
.Columns(col
 =>{
 col.Field("OrderID").Add();
}) | **Property:**
Toolbar

@(Html.EJ().Grid("Grid")
 .Toolbar(new List<string>() { "Edit"
 })
.Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Delete | **Property:** *ToolbarSettings.ToolbarItems*

 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>{
toolbar.ShowToolbar()
.ToolbarItems(items=>{
items.AddTool(ToolBarItems.Dele
 te);});
})
.Columns(col =>{
 col.Field("OrderID").Add();
}) | **Property:**
Toolbar

@(Html.EJ().Grid("Grid")
 .Toolbar(new List<string>() { "Delete"
 })
.Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Update | **Property:** *ToolbarSettings.ToolbarItems*

 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>{
toolbar.ShowToolbar()
.ToolbarItems(items=>{
items.AddTool(ToolBarItems.Upd
 ate);});
})
.Columns(col =>{
 col.Field("OrderID").Add();
}) | **Property:**
Toolbar

@(Html.EJ().Grid("Grid")
 .Toolbar(new List<string>() { "Update"
 })
.Columns(col =>{
 col.Field("OrderID").Add();
}).Render()

| Cancel | **Property:** *ToolbarSettings.ToolbarItems*
 @(Html.EJ().Grid<object>("Grid")

 .ToolbarSettings(toolbar
 =>
toolbar.ShowToolbar()
.ToolbarItems(items=>
items.AddTool(ToolBarItems.Canc
 el));
}
.Columns(col =>
 col.Field("OrderID").Add();
}
)| **Property:**
Toolbar
 @Html.EJS().Grid("Grid")
 .Toolbar(new List<string>() { "Cancel"
 })
.Columns(col =>
 col.Field("OrderID").Add();
}).Render()

| ExcelExport | **Property:** *ToolbarSettings.ToolbarItems*
 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>
toolbar.ShowToolbar()
.ToolbarItems(items=>
items.AddTool(ToolBarItems.Exce
 lExport));
}
.Columns(col =>
 col.Field("OrderID").Add();
}
)|
Property: *Toolbar*
 @Html.EJS().Grid("Grid")
 .Toolbar(new List<string>() {
 "ExcelExport" })
.Columns(col =>

 col.Field("OrderID").Add();
}).Render()

| WordExport | **Property:** *ToolbarSettings.ToolbarItems*
 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>
toolbar.ShowToolbar()
.ToolbarItems(items=>
items.AddTool(ToolBarItems.Wor
 dExport));
}
.Columns(col =>
 col.Field("OrderID").Add();
}
)| Not
 Applicable

| PdfExport | **Property:** *ToolbarSettings.ToolbarItems*
 @(Html.EJ().Grid<object>("Grid")
 .ToolbarSettings(toolbar
 =>
toolbar.ShowToolbar()
.ToolbarItems(items=>
items.AddTool(ToolBarItems.PdfE
 xport));
}
.Columns(col =>
 col.Field("OrderID").Add();
}
)| **Property:**
Toolbar
 @Html.EJS().Grid("Grid")
 .Toolbar(new List<string>() { "PdfExport"
 })
.Columns(col =>
 col.Field("OrderID").Add();
}).Render()

| Refresh Toolbar | **Method:** *refreshToolbar()*
 @(Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.refreshToolbar();
</script>| Not Applicable

| Triggers when toolbar item is clicked | **Event:** *ToolbarClick*
 @(Html.EJ().Grid<object>("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSou
 rce)
.ClientSideEvents(eve=>
eve.ToolbarClick("toolbarClick"));)
Script:
<script><b
 r>function toolbarClick(){
</script>| **Event:**
ToolbarClick
 @Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
View
 Bag.dataSource)
.ToolbarClick("toolbarClick")
.Render()
Script:
<script>
functi
 on toolbarClick(){
</script>

GridLines

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *GridLines*
 @(Html.EJ().Grid<object>("Grid")

 .GridLines(GridLines.Vertical)
.Columns(col
 =>
 col.Field("OrderID").Add();
}
)| **Property:**


```
GridLines<br><br>@Html.EJS().Grid("Grid")<br>&#160;.GridLines(Syncfusion.EJ2<br>.Grids.GridL
ine.Vertical).Columns(col =>{<br> &#160;&#160;col.Field("OrderID").Add();<br>}).Render()|
```

Templates

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *DetailsTemplate*

 @Html.EJ().Grid<object>("Grid")
 .DetailsTemplate("#detailsTemplate")
Script:
<script id="detailsTemplate" type="text/x-jsrender">
You can add template elements here
</script>| **Property:** *DetailTemplate*

@Html.EJS().Grid("Grid")
 .DetailTemplate("#detailtemplate")
.Render()
Script:
<script type="text/x-template" id="detailtemplate">
You can add template elements here
</script>

| Default | **Property:** *RowTemplate*

 @Html.EJ().Grid<object>("Grid")
 .RowTemplate("#templateData")
Script:
<script id="templateData" type="text/x-jsrender">
You can add template elements here
</script>| **Property:** *RowTemplate*

@Html.EJS().Grid("Grid")
 .RowTemplate("#rowtemplate")
.Render()
Script:
<script id="rowtemplate" type="text/x-template">
You can add template elements here
</script>

| Refresh Template | **Method:** *refreshTemplate()*

 @Html.EJ().Grid<object>("Grid").Datasource
((IEnumerable<object>)
ViewBag.data Source)
Script:
<script>
var gridObj = \$("#Grid").data("ejGrid");
gridObj.refreshTemplate();
</script>| Not Applicable

| Triggers when detail template row is clicked to collapse | **Event:** *DetailsCollapse*

 @Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSou ce)
.ClientSideEvents(eve =>{
eve.DetailsCollapse("detailsCollapse");})
Script:
<script>
function detailsCollapse({})
</script>| Not Applicable

| Triggers when detail template row is initialized | **Event:** *DetailsDataBound*

 @Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSou ce)
.ClientSideEvents(eve =>{
eve.DetailsDataBound("detailsDataBound");})
Script:
<script>
function detailsDataBound({})
</script>| Not Applicable

| Triggers when detail template
row is clicked to expand | **Event:** *DetailsExpand*

 @Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSou ce)
.ClientSideEvents(eve =>{
eve.DetailsExpand("detailsExpand");})
Script:
<script>
function detailsExpand({})
</script>| **Event:**

DetailDataBound

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.DetailDataBound("detailDataBound")
.Render()
Script:
<script>
function detailDataBound({})
</script>

| Triggers when refresh the template column elements in the Grid | **Event:** *TemplateRefresh*

 @Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSou


```
ce)<br>.ClientSideEvents(eve
=>{<br>eve.TemplateRefresh("templateRefresh");}))<br>Script:<br><script><br>function
templateRefresh(){<br></script>| Not Applicable
```

Row/Column Drag and Drop

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| Default | **Property:** *AllowRowDragAndDrop*

 @ (Html.EJ().Grid<object>("Grid")

 .AllowRowDragAndDrop())
.RowDropSettings(drop
=>
drop.RowDropMapper("RowDropHandler")
.DropTargetID("#DestGrid"))
.Column
s(col =>{
 col.Field("OrderID").Add();
})
| **Property:**
AllowRowDragAndDrop

@Html.EJS().Grid("Grid")
 .AllowRowDragAndDrop(tru
e)
.RowDropSettings(new Syncfusion.EJ2.Grids.GridRowDropSettings() { TargetID =
"RowDragDrop" })
.Columns(col =>{

 col.Field("OrderID").Add();
}).Render()

| Triggers when the row is
being dragged | **Event:** *RowDrag*

@ (Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
ce)
.ClientSideEvents(eve
=>{
eve.RowDrag("rowDrag");}))
Script:
<script>
function
rowDrag(){
</script>| **Event:**
RowDrag

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBa
g.dataSource)
.RowDrag("rowDrag")
.Render()
Script:
<script>
function
rowDrag(){
</script>

| Triggers when the row drag begins | **Event:** *RowDragStart*

@ (Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
ce)
.ClientSideEvents(eve
=>{
eve.RowDragStart("rowDragStart");}))
Script:
<script>
function
rowDragStart(){
</script>| **Event:**
RowDragStart

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
Vie
wBag.dataSource)
.RowDragStart("rowDragStart")
.Render()
Script:
<script>
f
unction rowDragStart(){
</script>

| Triggers when the row is dropped | **Event:** *RowDrop*

@ (Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
ce)
.ClientSideEvents(eve
=>{
eve.RowDrop("rowDrop");}))
Script:
<script>
function
rowDrop(){
</script>| **Event:**
RowDrop

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBa
g.dataSource)
.RowDrop("rowDrop")
.Render()
Script:
<script>
function
rowDrop(){
</script>

| Triggers before the row is being dropped | **Event:** *BeforeRowDrop*

@ (Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
ce)
.ClientSideEvents(eve

```
=>{<br>eve.BeforeRowDrop("beforeRowDrop");})<br>Script:<br><script><br>function  
beforeRowDrop(){<br></script> | Not Applicable
```

| Triggers when the
column is being dragged | **Event:** *columnDrag*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.ColumnDrag("columnDrag");})
Script:
<script>
function
 columnDrag(){
</script> | Not Applicable

| Triggers when the
column drag begins | **Event:** *columnDragStart*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.columnDragStart("columnDragStart");})
Script:
<script>
function
 columnDragStart(){
</script> | Not Applicable

| Triggers when the
column is dropped | **Event:** *ColumnDrop*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
 =>{
eve.ColumnDrop("columnDrop");})
Script:
<script>
function
 columnDrop(){
</script> | Not Applicable

Frozen Rows and Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Default | **Property:** *ScrollSettings.FrozenRows*

 @(Html.EJ().Grid<object>("Grid")
 .AllowScrolling())
.ScrollSettings(scroll => {
 scroll.Height(337)
.Width(700).FrozenRows(2)
.FrozenColumns(2); })
)| **Property:**
FrozenRows

@Html.EJS().Grid("Grid")
 .FrozenRows(2).FrozenColumns(1)
.Render()

| isFrozen | **Property:** *Columns.IsFrozen*

 @(Html.EJ().Grid<object>("Grid")
 .Columns(col =>{
 col.Field("OrderID").
IsFrozen(true).Add();
})
)|
Property: *Columns.IsFrozen*

@Html.EJS().Grid("Grid")
 .Columns(col =>{
 col.Field("OrderID").IsFrozen(true).Add();
}).Render()

ForeignKey

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Default | **Property:** *Columns.ForeignKeyValue*

 @(Html.EJ().Grid<object>("Grid")
.Columns(col
 =>{
 col.Field("OrderID").Add();
 col.Field("EmployeeID")
.ForeignKeyField("EmployeeID")
.ForeignKeyValue("FirstName")
.DataSource((IEnumerable<object>)
ViewBag.dataSource2).Add();
})
)| **Property:**
Columns.ForeignKeyValue

@Html.EJS().Grid("Grid")
 .Columns(col =>{
 col.Field("OrderID").Add();
 col.Field("CustomerID")
.ForeignKeyField("CustomerID")
.ForeignKeyValue(

```
"ContactName")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource2).Add();<br>}.Render()
```

Auto Wrap

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *AllowTextWrap*

 @(Html.EJ().Grid<object>("Grid")
 .AllowTextWrap().Columns(col =>
 col.Field("OrderID").Add();
})
)|

Property:

```
AllowTextWrap<br><br>@Html.EJS().Grid("Grid")<br>&#160;.AllowTextWrap().Columns(col
=><br>&#160;&#160;col.Field("OrderID").Add();<br>}.Render()|
```

| Both | **Property:** *AllowTextWrap*

 @(Html.EJ().Grid<object>("Grid")
 .AllowTextWrap()
.TextWrapSettings(wrap => {wrap.WrapMode(WrapMode.Both);
})
.Columns(col =>
 col.Field("OrderID").Add();
})
)| **Property:**

```
AllowTextWrap<br><br>@Html.EJS().Grid("Grid")<br>&#160;.AllowTextWrap()<br>.TextWrapSet
tings(text =>
```

```
{<br>text.WrapMode<br>(Syncfusion.EJ2.Grids.WrapMode.Both);}<br>.Columns(col =><br>&#160;&#160;col.Field("OrderID").Add();<br>}.Render()|
```

| Header | **Property:** *AllowTextWrap*

 @(Html.EJ().Grid<object>("Grid")
 .AllowTextWrap()
.TextWrapSettings(wrap => {wrap.WrapMode(WrapMode.Header);
})
.Columns(col =>
 col.Field("OrderID").Add();
})
)| **Property:**

```
AllowTextWrap<br><br>@Html.EJS().Grid("Grid")<br>&#160;.AllowTextWrap()<br>.TextWrapSet
tings(text =>
```

```
{<br>text.WrapMode<br>(Syncfusion.EJ2.Grids.WrapMode.Header);}<br>.Columns(col =><br>&#160;&#160;col.Field("OrderID").Add();<br>}.Render()|
```

| Content | **Property:** *AllowTextWrap*

 @(Html.EJ().Grid<object>("Grid")
 .AllowTextWrap()
.TextWrapSettings(wrap =>
{wrap.WrapMode(WrapMode.Content); })
.Columns(col
=>
 col.Field("OrderID").Add();
})
)| **Property:**

```
AllowTextWrap<br><br>@Html.EJS().Grid("Grid")<br>&#160;.AllowTextWrap()<br>.TextWrapSet
tings(text =>
```

```
{<br>text.WrapMode<br>(Syncfusion.EJ2.Grids.WrapMode.Content);}<br>.Columns(col =><br>&#160;&#160;col.Field("OrderID").Add();<br>}.Render()|
```

Responsive

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *IsResponsive*

 @(Html.EJ().Grid<object>("Grid")
 .IsResponsive(true)
.EnableResponsiveRow(true)
.Columns(col
=>
 col.Field("OrderID").Add();
})
)| Not Applicable

State Persistence

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Default | **Property:** *EnablePersistence*
 @(Html.EJ().Grid<object>("Grid")

 .EnablePersistence())
.Columns(col
 =>{
 col.Field("OrderID").Add();
})
| **Property:**
EnablePersistence
@Html.EJS().Grid("Grid")
 .EnablePersistence()
.Column
 s(col =>{
 col.Field("OrderID").Add();
}).Render()|

Right to Left - RTL

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Default | **Property:** *EnableRtl*
 @(Html.EJ().Grid<object>("Grid")

 .EnableRtl())
.Columns(col =>{
 col.Field("OrderID").Add();
})
| **Property:** *EnableRtl*
 @Html.EJS().Grid("Grid")
 .EnableRtl()
.Columns(col
 =>{
 col.Field("OrderID").Add();
}).Render()|

ToolTip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Default | **Property:** *ClipMode*
 @(Html.EJ().Grid<object>("Grid")

 .Columns(col
 =>{
col.Field("OrderID")
.ClipMode(ClipMode.Clip).Add();
col.Field("CustomerID")

 >.ClipMode(ClipMode.Ellipsis).Add();
col.Field("ShipCity")
.ClipMode(ClipMode.EllipsisWi
 thTooltip).Add();
})
| **Property:**
ClipMode
@Html.EJS().Grid("Grid")
 .EnableRtl()
.Columns(col
 =>{
col.Field("OrderID")
.ClipMode(Syncfusion.EJ2.Grids
.ClipMode.Clip)
.Add();

 r>col.Field("CustomerID")
.ClipMode(Syncfusion.EJ2.Grids
.ClipMode.Ellipsis)
.Add();

col.Field("ShipCity")
.ClipMode(Syncfusion.EJ2.Grids
.ClipMode.EllipsisWithTooltip)<
 br>.Add();
}).Render()|

Aggregate/Summary

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Footer Aggregate | **Property:** *ShowSummary*
 @(Html.EJ().Grid<object>("Grid")

 .ShowSummary()
.SummaryRow(row
 =>{
row.Title("Sum")
.SummaryColumns(col =>
 {
col.SummaryType(SummaryType.Sum)
.Format("{0:C}")
.DisplayColumn("Freight")<
 br>.DataMember("Freight").Add();
}).Add();
})
| **Property:**
Aggregates
@Html.EJS().Grid("Grid")
 .Aggregates(agg=>{
agg.Columns(n
 ew List<Syncfusion.EJ2
.Grids.GridAggregateColumn>())
new
 Syncfusion.EJ2
.Grids.GridAggregateColumn() {
Field = "Freight", Format = "C2",
Type
 = "Sum", FooterTemplate = "Sum: \${Sum}" } }).Add();
}).Render()|

| Caption Aggregate | **Property:** *ShowSummary*
 @(Html.EJ().Grid<object>("Grid")

 .ShowSummary()
.SummaryRow(row

```
=>{<br>row.ShowTotalSummary(false)<br>.ShowCaptionSummary(true)<br>.SummaryColumns(
col
=>{<br>col.SummaryType(SummaryType.Average)<br>.Format("{0:C2}")<br>.DisplayColumn("Fr
eight")<br>.DataMember("Freight")<br>.Prefix("Average").Add();<br>}}.Add();<br>}} | Property:
Aggregates<br><br>@Html.EJS().Grid("Grid")<br>#160;.Aggregates(agg=>{<br>agg.Columns(n
ew List<Syncfusion.EJ2<br>.Grids.GridAggregateColumn>())<br>new
Syncfusion.EJ2<br>.Grids.GridAggregateColumn() {<br>Field = "Freight", Format = "C2",<br>Type
= "Sum", GroupCaptionTemplate = "Sum: ${Sum}" } }).Add();<br>}).Render();
```

| Get Summary values | **Method:** *getSummaryValues(summaryCol, summaryData)*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getSummaryValues(summaryCol,
 window.gridData);
</script> | Not Applicable

Grid Export

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Adds a grid model property
 which is to be ignored on exporting grid | **Method:**

addIgnoreOnExport(propertyNames)

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.addIgnoreOnExport
("filterSettings");
</script> |
 Not Applicable |

Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Add or Remove Columns | **Method:** *columns(columnDetails, [action])()*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.columns("OrderID",
 "remove");
gridObj.columns("CustomerID", "add");
</script> | **Property:**
columns

Grid is initially rendered with OrderID and CustomerID columns.
Then if you want
 to add ShipAddress column, you have to reset the value for column property as gridObj.columns =
 [{field:"OrderID"}, {field:"CustomerID"}, {field:"ShipAddress"}]; Then to remove the
 CustomerID column, reset the column property as, gridObj.columns = [{field:"OrderID"},
 {field:"ShipAddress"}];

| Get Column By Field | **Method:** *getColumnByField(fieldName)*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getColumnByField("OrderID");
</script> | **Method:**
getColumnByField(fieldName)

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<o
 bject>
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =

```
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getColumnByField("OrderID")<br></script>
```

| Get Column By HeaderText | **Method:** `getColumnByHeaderText(headerText)`

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getColumnByHeaderText
("Order ID");
</script> |
 Column object for a corresponding headerText can be able to get by iterating the gridObj.columns with the
 headerText

| Get Column By Index | **Method:** `getColumnByIndex(columnIndex)`

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getColumnByIndex(1);
</script> | **Method:**
`getColumnByIndex(columnIndex)`

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable
 <object>
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.getColumnByIndex(1)
</
 script>

| Get Column Fieldnames | **Method:** `getColumnFieldNames()`

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getColumnFieldNames();
</script> | **Method:**
`getColumnFieldNames()`

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object
 >
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.getColumnFieldNames()

 </script>

| Get Column Index By Field | **Method:** `getColumnIndexByField(fieldName)`

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getColumnIndexByField("OrderID");
</script> |
Method:
`getColumnIndexByField(fieldName)`

@Html.EJS().Grid("Grid")
.DataSource((IEnum
 erable<object>
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.getColumnIndexByField("Or
 derID");
</script>

| Get Column Index By headerText | **Method:** `getColumnIndexByHeaderText(headerText,
 [field])`

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getColumnIndexByHeaderText("Order ID");
</script> |
 The Column Index for a corresponding headerText can be get by iterating the gridObj.columns with the
 headerText

| Set Width to columns | **Method:** `setWidthToColumns()`

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>
ViewBag.data

Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.setWidthToColumns();
</script>| **Method:**

widthService.setWidthToColumns()
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.widthService.setWidthToColumns()
</script>

| Get HeaderText By FieldName | **Method:** *getHeaderTextByFieldName()*

@ (Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getHeaderTextByFieldName("OrderID");
</script>|
Not Applicable

| Get Hidden Columnname | **Method:** *getHiddenColumnNames()*

@ (Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getHiddenColumnNames();
</script>| Not Applicable

| Get Visible Columns/Names | **Method:** *getVisibleColumnNames()*

@ (Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getVisibleColumnNames();
</script>| **Method:**
getVisibleColumns()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.getVisibleColumns();
</script>

| Select Columns | **Method:** *selectColumns(fromIndex)*

@ (Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.selectColumns(1,4);
</script>| Not Applicable

| Select Specific Columns based on Index | **Method:**

selectColumns(columnIndex,[toIndex])

@ (Html.EJ().Grid<object>
("Grid").DataSource
((IEnumerable<object>)
ViewBag.dataSource))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.selectColumns(1,4);
</script>| Not Applicable

Row

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| EnableRowHover | **Property:** *EnableRowHover*

@ (Html.EJ().Grid<object>("Grid")
 .EnableRowHover()
.Columns(col
=>
 col.Field("OrderID").Add();
)}
)| **Property:**
EnableHover

@Html.EJS().Grid("Grid")
 .EnableHover()
.Columns(col
=>
 col.Field("OrderID").Add();
}).Render()|

| Get Row Height | **Method:** *getRowHeight()*
 @(Html.EJ().Grid<object>(
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getRowHeight();
</script>| **Method:**
getRowHeight()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById(
('Grid')).ej2_instances[0];
gridObj.getRowHeight();
</scrip
 t>

| Refresh Row Height | **Method:** *rowHeightRefresh()*
 @(Html.EJ().Grid<object>(
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.rowHeightRefresh();
</script>| Not Applicable

| Get index by Row Element | **Method:** *getIndexByRow(\$tr)*
 @(Html.EJ().Grid<object>(
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getIndexByRow(\$("#.gridcontent tr").first());
</script>|
 Not Applicable

| Get Row by its Index | **Method:** *getRowByIndex(from, to)*
 @(Html.EJ().Grid<object>(
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getRowByIndex(3, 6);
</script>| **Method:**
getRowByIndex(index)

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById(
('Grid')).ej2_instances[0];
gridObj.getRowByIndex(1);
</sc
 ript>

| Get rendered rows | **Method:** *getRows()*
 @(Html.EJ().Grid<object>(
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getRows();
</script>| **Method:**
getRows()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewB
 ag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById(
('Grid')).ej2_instances[0];
gridObj.getRows();
</script>

| Triggers while hover the grid row | **Event:** *RowHover*
 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
 ce)
.ClientSideEvents(eve
 =>{
eve.RowHover("rowHover");}))
Script:
<script>
function
 rowHover(){
</script>| Not Applicable

[Show/Hide Columns](#)

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| Hide Columns by using method | **Method:** *hideColumns(headerText)*
 @(Html.EJ().Grid<object>("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource)
Script:
 \$("#Grid").data("ejGrid");
gridObj.hideColumns("Order ID");
Method:
hideColumns(headerText)
 @Html.EJS().Grid("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource).Render()
Script:
 var gridObj = document.getElementById('Grid').ej2_instances[0];
gridObj.hideColumns("Order ID");

| Show Columns by using method | **Method:** *showColumns(headerText)*
 @(Html.EJ().Grid<object>("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource)
Script:
 \$("#Grid").data("ejGrid");
gridObj.showColumns("Order ID");
Method:
showColumns(headerText)
 @Html.EJS().Grid("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource).Render()
Script:
 var gridObj = document.getElementById('Grid').ej2_instances[0];
gridObj.showColumns("Order ID");

Column Chooser

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *ShowColumnChooser*
 @(Html.EJ().Grid<object>("Grid").ShowColumnChooser() .Columns(col =>{
 .col.Field("OrderID").ShowInColumnChooser(false).Add();
col.Field("Freight").Add();
})
| **Property:**
ShowColumnChooser
 @Html.EJS().Grid("Grid").ShowColumnChooser()
.Columns(col =>{
 .col.Field("OrderID").ShowInColumnChooser(false).Add();
col.Field("City").Add();
}).Render()|

Header

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Refresh Header | **Method:** *refreshHeader()*
 @(Html.EJ().Grid<object>("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource)
Script:
 \$("#Grid").data("ejGrid");
gridObj.refreshHeader();
Method:
refreshHeader()
 @Html.EJS().Grid("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource).Render()
Script:
 var gridObj = document.getElementById('Grid').ej2_instances[0];
gridObj.refreshHeader();

| Triggers every time a request is made to access particular header cell information, element and data. |

Event: *MergeHeaderCellInfo*

@(Html.EJ().Grid<object>("Grid").DataSource<IEnumerable<object>>()>ViewBag.dataSource

```
ce)<br>.ClientSideEvents(eve
=>{<br>eve.MergeHeaderCellInfo("mergeHeaderCellInfo");}))<br>Script:<br><script><br>function
mergeHeaderCellInfo(){<br></script> | Not Applicable
```

| Triggers every time a request is made to access particular cell information, element and data | **Event:**
MergeCellInfo
</br>

```
@(Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSou
ce)<br>.ClientSideEvents(eve
=>{<br>eve.MergeCellInfo("mergeCellInfo");}))<br>Script:<br><script><br>function
mergeCellInfo(){<br></script> | Not Applicable
```

DataSource

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To set a new DataSource Dynamically by using the method | **Method:**

```
dataSource(datasource,[templateRefresh])<br></br>
@(<Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.dataSource(newdataSource);<br></script> | Property:
dataSource<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>View
Bag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.datasource =
newdataSource<br></script>
```

Print

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Print the grid | **Method:** *print()*
</br>

```
@(<Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object>)<br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.print();<br></script> | Method:
print()<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.d
ataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.print();<br></script>
```

| Triggers before printing the grid | **Event:** *BeforePrint*
</br>

```
@(<Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSou
ce)<br>.ClientSideEvents(eve
=>{<br>eve.BeforePrint("beforePrint");}))<br>Script:<br><script><br>function
beforePrint(){<br></script> | Event:
BeforePrint<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>View
Bag.dataSource)<br>.BeforePrint("beforePrint")<br>.Render()<br>Script:<br><script><br>functio
n beforePrint(){<br></script>
```

Scrolling

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

|Get ScrollObject | **Method:** *getScrollObject()*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getScrollObject();<br></script>| Property:
grid.scrollModule<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object><br>
ViewBag.dataSource)<br>.Render())<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>var scrollObj =
gridObj.scrollModule;<br></script>
```

PrimaryKey

|Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

|Default | **Property:** *Columns.IsPrimaryKey*

 @Html.EJ().Grid<object>("Grid")

 .Columns(col
=>{
 col.Field("OrderID")
.IsPrimaryKey(true).Add();
col.Field("Freight").Add();

})
| **Property:** *Columns.IsPrimaryKey*

@Html.EJS().Grid("Grid")
.Columns(col
=>{

 col.Field("OrderID")
.IsPrimaryKey(true)
.Add();
col.Field("City").Add();

}).Render()|

|Get the PrimaryKey fieldnames | **Method:** *getPrimaryKeyFieldNames()*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getPrimaryKeyFieldNames();<br></script>| Method:
<br>getPrimaryKeyFieldNames()<br><br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable
<object><br>ViewBag.dataSource)<br>.Render())<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getPrimaryKeyFieldNames()
;<br></script>
```

Grid

|Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

|Get the Browser Details| **Method:** *getBrowserDetails()*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getBrowserDetails();<br></script>| In Essential JS 2, it can
be <br>achieved by using Browserclass of ej2-base
```

|Set dimension for the grid | **Method:** *setDimension(height, width)*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
Source))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.setDimension(300,400);<br></script>| Not Applicable
```

|set maximum width for mobile | **Method:** *setPhoneModeMaxWidth(value)*


```
@(Html.EJ().Grid<object><br>("Grid").Datasource<br>((IEnumerable<object><br>ViewBag.data
```

Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.setPhoneModeMaxWidth(500);
</script> | Not
Applicable

| Render the grid | **Method:** *render()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.render();
</script> | **Method:**
render()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag
.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.render();
</script>

| Reset the model collections like pageSettings,
groupSettings, filterSettings, sortSettings and
summaryRows. | **Method:** *resetModelCollections()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.resetModelCollections();
</script> | Not Applicable

| Destroy the grid | **Method:** *destroy()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.destroy();
</script> | **Method:**
destroy()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBa
g.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.destroy();
</script>

| Get Content Element | **Method:** *getContent()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getContent();
</script> | **Method:**
getContent()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
Vie
wBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.getContent();
</script>

| Get Content Table | **Method:** *getContentTable()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getContentTable();
</script> | **Method:**
getContentTable()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)

ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
document.getElementById
('Grid').ej2_instances[0];
gridObj.getContentTable();
</scr
ipt>

| Get Current View Data | **Method:** *getCurrentViewData()*

@(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
Source))
Script:
<script>
var gridObj =
\$("#Grid").data("ejGrid");
gridObj.getCurrentViewData();
</script> | **Method:**
getCurrentViewRecords()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<objec

```
t>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getCurrentViewRecords();<br></script>
```

```
| Get Data Row | Method: getDataByIndex(rowIndex)<br></br>
@(<Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.dataSource))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getDataByIndex(0);<br></script> | Method:
getDataRows()<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getDataRows();<br></script>
>
```

```
| Get Fieldname by using the headertext | Method:
<br>getFieldNameByHeaderText(headerText)<br></br>
@(<Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.dataSource))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getFieldNameByHeaderText<br>("Order
ID");<br></script> | Not Applicable
```

```
| Get FooterContent | Method: getFooterContent()<br></br>
@(<Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.dataSource))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getFooterContent();<br></script> | Method:
getFooterContent()<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getFooterContent();<br></script>
```

```
| Get FooterContent Table | Method: getFooterTable()<br></br>
@(<Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.dataSource))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getFooterTable();<br></script> | Method:
getFooterContentTable()<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getFooterContentTable();<br></script>
```

```
| Get HeaderContent | Method: getHeaderContent()<br></br>
@(<Html.EJ().Grid<object><br>("Grid").DataSource<br>((IEnumerable<object>)<br>ViewBag.dataSource))<br>Script:<br><script><br>var gridObj =
$("#Grid").data("ejGrid");<br>gridObj.getHeaderContent();<br></script> | Method:
getHeaderContent()<br></br>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.dataSource)<br>.Render()<br>Script:<br><script><br>var gridObj =
document.getElementById<br>('Grid').ej2_instances[0];<br>gridObj.getHeaderContent();<br></script>
```

| Get HeaderContent Table | **Method:** *getHeaderTable()*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.getHeaderTable();
</script> | **Method:**
getHeaderTable()

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)

 ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.getHeaderTable();
</scri
 pt>

| Refresh Content | **Method:** *refreshContent()*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.refreshContent();
</script> | **Method:**

contentModule.refreshContentRows()

@Html.EJS().Grid("Grid")
.DataSource((IEnum
 erable<object>)
ViewBag.dataSource)
.Render()
Script:
<script>
var gridObj =
 document.getElementById
('Grid').ej2_instances[0];
gridObj.
contentModule.refresh
 ContentRows();
</script>

| Refresh Data | **Method:** *refreshData()*

 @(Html.EJ().Grid<object>
("Grid").Datasource
((IEnumerable<object>)
ViewBag.data
 Source))
Script:
<script>
var gridObj =
 \$("#Grid").data("ejGrid");
gridObj.refreshData();
</script> | Not Applicable

| Triggers every time a request is
made to access particular cell
information, element and data |
Event: *QueryCellInfo*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
 ce)
.ClientSideEvents(eve
 =>{
eve.QueryCellInfo("queryCellInfo");}))
Script:
<script>
function
 queryCellInfo(){
</script> | **Event:**
QueryCellInfo

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
Vie
 wBag.dataSource)
.QueryCellInfo("queryCellInfo")
.Render()
Script:
<script>
fu
 nction queryCellInfo(){
</script>

| Triggers every time a request is
made to access row information,
element and data | **Event:**
RowDataBound

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
 ce)
.ClientSideEvents(eve
 =>{
eve.RowDataBound("rowDataBound");}))
Script:
<script>
function
 rowDataBound(){
</script> | **Event:**
RowDataBound

@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
Vi
 ewBag.dataSource)
.RowDataBound("rowDataBound")
.Render()
Script:
<script><
 br/>function rowDataBound(){
</script>

| Triggers for every grid
action before it get started | **Event:** *ActionBegin*

 @(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSour
 ce)
.ClientSideEvents(eve
 =>{
eve.ActionBegin("actionBegin");}))
Script:
<script>
function

`actionBegin(){}
</script>| Event:`

`ActionBegin
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.ActionBegin("actionBegin")
.Render()
Script:
<script>
function actionBegin(){}
</script>`

| Triggers for every grid
action success event| **Event:** `ActionComplete
</br>`

`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.ActionComplete("actionComplete");}))
Script:
<script>
function
actionComplete(){}
</script>| Event:`

`ActionComplete
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.ActionComplete("actionComplete")
.Render()
Script:
<script>
>
function actionComplete(){}
</script>`

| Triggers for every grid
server failure event| **Event:** `ActionFailure
</br>`

`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.ActionFailure("actionFailure");}))
Script:
<script>
function
actionFailure(){}
</script>| Event:`

`ActionFailure
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.ActionFailure("actionFailure")
.Render()
Script:
<script>
function
actionFailure(){}
</script>`

| Triggers when the grid is bound
with data during rendering| **Event:** `DataBound
</br>`

`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.DataBound("dataBound");}))
Script:
<script>
function
dataBound(){}
</script>| Event:`

`DataBound
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.DataBound("dataBound")
.Render()
Script:
<script>
function
dataBound(){}
</script>`

| Triggers when the grid is
going to destroy| **Event:** `Destroy
</br>`

`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve
=>{
eve.Destroy("destroy");}))
Script:
<script>
function destroy(){}
</script>|`

Event:

`Destroyed
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)
.Destroyed("destroyed")
.Render()
Script:
<script>
function
destroyed(){}
</script>`

| Triggers when initial load of the grid| **Event:** `Load
</br>`

`@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)
ViewBag.dataSource)
.ClientSideEvents(eve =>{
eve.Load("load");}))
Script:
<script>
function
load(){}
</script>| Event:`

`Load
</br>@Html.EJS().Grid("Grid")
.DataSource((IEnumerable<object>)
ViewBag.dataSource)`

```
taSource)<br>.Load("load")<br>.Render()<br>Script:<br><script><br>function
load(){}<br></script>
```

| Triggers when the grid is rendered completely | **Event:** *Create*


```
@(Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve =>{<br>eve.Create("create");}))<br>Script:<br><script><br>function
create(){}<br></script> | Event:
Created<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>ViewBag.
dataSource)<br>.Created("created")<br>.Render()<br>Script:<br><script><br>function
created(){}<br></script>
```

| Triggers when the record is clicked | **Event:** *RecordClick*


```
@(Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.RecordClick("recordClick");}))<br>Script:<br><script><br>function
recordClick(){}<br></script> | Not Applicable
```

| Triggers when right clicked on grid element | **Event:** *RightClick*


```
@(Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.RightClick("rightClick");}))<br>Script:<br><script><br>function
rightClick(){}<br></script> | Not Applicable
```

| Triggers when the
record is double clicked | **Event:** *RecordDoubleClick*


```
@(Html.EJ().Grid<object>("Grid")<br>.Datasource((IEnumerable<object>)<br>ViewBag.dataSour
ce)<br>.ClientSideEvents(eve
=>{<br>eve.RecordDoubleClick<br>("recordDoubleClick");}))<br>Script:<br><script><br>function
recordDoubleClick(){}<br></script> | Event:
RecordDoubleClick<br/><br/>@Html.EJS().Grid("Grid")<br>.DataSource((IEnumerable<object>)<br>
ViewBag.dataSource)<br>.RecordDoubleClick<br>("recordDoubleClick")<br>.Render()<br>Script
:<br><script><br>function recordDoubleClick(){}<br></script>
```

Enable/Disable Grid and its actions

You can enable/disable the Grid and its actions by applying/removing corresponding CSS styles.

To enable/disable the grid and its actions, follow the given steps:

Step 1: Create CSS class with custom style to override the default style of Grid.

```
`css
.disablegrid {
pointer-events: none;
opacity: 0.4;
}
.wrapper {
cursor: not-allowed;
}
```


Step 2: Add/Remove the custom CSS class to the Grid in the click event handler of Button.

```
`typescript
<script>
document.getElementById('element').onclick = function () {
var gridInst = document.getElementById("Grid").ej2_instances[0];
if (gridInst.element.classList.contains('disablegrid')) {
gridInst.element.classList.remove('disablegrid');
document.getElementById("GridParent").classList.remove('wrapper');
}
else {
gridInst.element.classList.add('disablegrid');
document.getElementById("GridParent").classList.add('wrapper');
}
}
</script>
```

In the below demo, the button click will enable/disable the Grid and its actions.

CSHTML

```
@Html.EJS().Button("element").Content("Enable/Disable Grid").Render()
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").Add();
col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel" }).Render()
<style>
.disablegrid {
pointer-events: none;
opacity: 0.4;
}
.wrapper {
cursor: not-allowed;
}
</style>
<script>
```

```

document.getElementById('element').onclick = function () {
    var gridInst = document.getElementById("Grid").ej2_instances[0];
    if (gridInst.element.classList.contains('disablegrid')) {
        gridInst.element.classList.remove('disablegrid');
    }
    document.getElementById("GridParent").classList.remove('wrapper');
}
else {
    gridInst.element.classList.add('disablegrid');
    document.getElementById("GridParent").classList.add('wrapper');
}
}
</script>

```

DISABLEGRID.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Print the expanded state from other pages

By default, the expanded child grids will be printed from the current page alone. You can print the expanded child grids from other pages by using the [Load](#) and [ActionBegin](#) event.

In the following example, we have printed expanded child grids from other pages.

CSHTML

```

@{
    var SecondChildGrid = new Syncfusion.EJ2.Grids.Grid() { DataSource =
    (IEnumerable<object>)ViewBag.CustomerDataSource,
        QueryString = "CustomerID",
        Columns = new List<Syncfusion.EJ2.Grids.GridColumn>
        {
            new Syncfusion.EJ2.Grids.GridColumn() { Field="CustomerID",
            HeaderText="Customer ID", Width="90",
            TextAlign=Syncfusion.EJ2.Grids.TextAlign.Right },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="ShipCountry",
            HeaderText="Ship Country", Width="90" },
        }
    };
    var ChildGrid = new Syncfusion.EJ2.Grids.Grid() { DataSource =
    (IEnumerable<object>)ViewBag.DataSource,
        QueryString = "EmployeeID",
        Columns = new List<Syncfusion.EJ2.Grids.GridColumn>
        {
            new Syncfusion.EJ2.Grids.GridColumn() { Field="OrderID",
            HeaderText="Order ID", Width="120" },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="Freight",
            HeaderText="Freight", Width="120", Format="C2",
            TextAlign=Syncfusion.EJ2.Grids.TextAlign.Right },
            new Syncfusion.EJ2.Grids.GridColumn() { Field="ShipName",
            HeaderText="Ship Name", Width="150" },
        }
    };
}

```

```

        new Syncfusion.EJ2.Grids.GridColumn() { Field="ShipCity",
HeaderText="Ship City", Width="120" },
    },
    ChildGrid = SecondChildGrid
};
}

@Html.EJS().Grid("HierarchyPrint").DataSource((IEnumerable<object>)ViewBag.E
mpDataSource).HierarchyPrintMode(Syncfusion.EJ2.Grids.HierarchyGridPrintMode
.Expanded).Columns(col =>
{
    col.Field("EmployeeID").HeaderText("Employee
ID").Width("125").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("FirstName").HeaderText("Name").Width("125").Add();
    col.Field("Title").HeaderText("Title").Width("180").Add();
    col.Field("City").HeaderText("City").Width("135").Add();
}).AllowSorting().AllowPaging().PageSettings(page =>
page.PageSize(5)).Toolbar(new List<string>() {
"Print"}).ChildGrid(ChildGrid).ActionBegin("actionBegin").Load("load").Rende
r()

<script>
    var expandedChildGrid = {};
    function actionBegin(args) {
        if (args.requestType === 'paging') {
            expandedChildGrid = ej.base.extend({}, expandedChildGrid,
getExpandedState(this, 'Expanded', args.previousPage));
        }
    }
    function load() {
        this.on(ej.grids.printGridInit, printInit, this);
    }
    function printInit(gridModel) {
        gridModel.printgrid.expandedRows = ej.base.extend({},
expandedChildGrid, gridModel.expandedRows);
    }
    function getExpandedState(gObj, hierarchyPrintMode, currentPage) {
        var obj = {};
        var rows = gObj.getRowsObject();
        for (var i = 0; i < rows.length; i++) {
            var row = rows[i];
            if (row.isExpand && !row.isDetailRow) {
                var index = gObj.allowPaging ? row.index + (currentPage *
gObj.pageSettings.pageSize) - gObj.pageSettings.pageSize : row.index;
                obj[index] = {};
                obj[index].isExpand = true;
                obj[index].gridModel =
ej.grids.getPrintGridModel(row.childGrid, hierarchyPrintMode);
            }
        }
        return obj;
    }
}
</script>

```

HIERARCHYPRINT.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrdersDetails.GetAllRecords();
    ViewBag.EmpDataSource = EmployeeView.GetAllRecords();
    ViewBag.CustomerDataSource = Customer.GetAllRecords();
    return View();
}
```

Perform CRUD operation using anti-forgery token

Anti-forgery token is used between the client and server to prevent cross-site request forgery (CSRF) attack. For more information on preventing CSRF attack, refer to the [link](#).

While performing grid save operation, you can send anti-forgery token along with the save request using the below custom adaptor.

This custom adaptor will read the anti-forgery token from the hidden element and send it along with the request. Also content type is set to **application/x-www-form-urlencoded; charset=UTF-8** since the **ValidateAntiForgeryToken** will look for the token in the form encoded data.

```
`javascript
<script>
window.customAdaptor = new ej.data.UrlAdaptor();
customAdaptor = ej.base.extend(customAdaptor, {
processResponse: function (data, ds, query, xhr, request, changes) {
request.data = JSON.stringify(data);
return ej.data.UrlAdaptor.prototype.processResponse.call(this, data, ds, query, xhr, request, changes);
},
insert: function (dm, data, tableName) {
return {
url: dm.dataSource.insertUrl || dm.dataSource.crudUrl || dm.dataSource.url,
data: $.param({
//Added the anti-forgery token.
RequestVerificationToken: document.getElementsByName("RequestVerificationToken")[0].value,
value: data,
table: tableName,
action: 'insert'
}),
contentType: 'application/x-www-form-urlencoded; charset=UTF-8'
}
},
update: function (dm, keyField, value, tableName) {
```

```

return {
url: dm.dataSource.updateUrl || dm.dataSource.crudUrl || dm.dataSource.url,
data: $.param({
//Added the anti-forgery token.
RequestVerificationToken: document.getElementsByName("RequestVerificationToken")[0].value,
value: value,
table: tableName,
action: 'insert'
}),
contentType: 'application/x-www-form-urlencoded; charset=UTF-8'
};
},
});
</script>
`

```

Now assign the custom adaptor to the grid as follows.

```

`javascript
<script>
function load(args) {
this.dataSource.adaptor = customAdaptor;
}
</script>
`

```

CSHTML

```

@Html.EJS().Grid("RemoteSaveAdaptor").DataSource(dataManager => {
dataManager.Json(ViewBag.dataSource.ToArray()).InsertUrl("/Home/Insert").RemoveUrl("/Home/Delete").UpdateUrl("/Home/Update").Adaptor("RemoteSaveAdaptor");
}).EditSettings(e => {
e.AllowAdding(true).AllowEditing(true).AllowDeleting(true); }).Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Width("120").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width("150").Add();

col.Field("Freight").HeaderText("Freight").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("C2").Add();

```

```

        col.Field("OrderDate").HeaderText("Order
Date").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();
    }).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel"
    }).Render()
    <script type="text/javascript">
        window.customAdaptor = new ej.data.RemoteSaveAdaptor();
        customAdaptor = ej.base.extend(customAdaptor, {
            insert(dm, data, tableName) {
                this.updateType = 'add';
                return {
                    url: dm.dataSource.insertUrl || dm.dataSource.crudUrl ||
dm.dataSource.url,
                    data: $.param({
                        //Added the anti-forgery token.
                        __RequestVerificationToken:
document.getElementsByName("__RequestVerificationToken")[0].value,
                        value: data,
                        table: tableName,
                        action: 'insert'
                    }),
                    contentType: 'application/x-www-form-urlencoded;
charset=UTF-8'
                };
            },
            update(dm, keyField, value, tableName) {
                debugger;
                this.updateType = 'update';
                this.updateKey = keyField;
                return {
                    type: 'POST',
                    url: dm.dataSource.updateUrl || dm.dataSource.crudUrl ||
dm.dataSource.url,
                    data: $.param({
                        //Added the anti-forgery token.
                        __RequestVerificationToken:
document.getElementsByName("__RequestVerificationToken")[0].value,
                        value: value,
                        action: 'update',
                        keyColumn: keyField,
                        key: value[keyField],
                        table: tableName
                    }),
                    contentType: 'application/x-www-form-urlencoded;
charset=UTF-8'
                };
            },
            remove(dm, keyField, value, tableName) {
                ej.data.JsonAdaptor.prototype.remove(dm, keyField, value);
                return {
                    type: 'POST',
                    url: dm.dataSource.removeUrl || dm.dataSource.crudUrl ||
dm.dataSource.url,
                    data: $.param({
                        //Added the anti-forgery token.
                        __RequestVerificationToken:
document.getElementsByName("__RequestVerificationToken")[0].value,

```

```

                key: value,
                keyColumn: keyField,
                table: tableName,
                action: 'remove'
            )),
            contentType: 'application/x-www-form-urlencoded;
charset=UTF-8'
        });
    }
});
function load(args) {
    this.dataSource.adaptor = customAdaptor;
}
</script>

```

ANTI-FORGERY-TOKEN.CS

```

public class HomeController : Controller
{
    public static List<Orders> order = new List<Orders>();

    public ActionResult Index()
    {
        if (order.Count == 0)
            BindDataSource();
        ViewBag.data = order.ToArray();
        return View();
    }
    public void BindDataSource()
    {
        int code = 10000;
        for (int i = 1; i < 10; i++)
        {
            order.Add(new Orders(code + 1, "VINET", i + 0, 2.3 * i, new
DateTime(1991, 05, 15), "Berlin"));
            order.Add(new Orders(code + 2, "ANATR", i + 2, 3.3 * i, new
DateTime(1990, 04, 04), "Madrid"));
            order.Add(new Orders(code + 3, "ANTON", i + 1, 4.3 * i, new
DateTime(1957, 11, 30), "Cholchester"));
            order.Add(new Orders(code + 4, "BLONP", i + 3, 5.3 * i, new
DateTime(1930, 10, 22), "Marseille"));
            order.Add(new Orders(code + 5, "BOLID", i + 4, 6.3 * i, new
DateTime(1953, 02, 18), "Tsawassen"));
            code += 5;
        }
    }
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Update(Orders value, string action)
    {
        var data = order.Where(or => or.OrderID ==
value.OrderID).FirstOrDefault();
        if (data != null)
        {
            data.OrderID = value.OrderID;
            data.CustomerID = value.CustomerID;

```

```

        data.EmployeeID = value.EmployeeID;
        data.Freight = value.Freight;
        data.OrderDate = value.OrderDate;
        data.ShipCity = value.ShipCity;
    }
    return Json(value);
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Insert(Orders value, string action)
{
    order.Insert(0, value);
    return Json(value);
}
[HttpPost]
[ValidateAntiForgeryToken]
public void Remove(int key)
{
    var data = order.Where(or => or.OrderID == key).FirstOrDefault();
    if (data != null)
    {
        order.Remove(data);
    }
}
public class Orders
{
    public Orders()
    {
    }
    public Orders(long OrderId, string CustomerId, int EmployeeId,
double Freight, DateTime OrderDate, string ShipCity)
    {
        this.OrderID = OrderId;
        this.CustomerID = CustomerId;
        this.EmployeeID = EmployeeId;
        this.Freight = Freight;
        this.OrderDate = OrderDate;
        this.ShipCity = ShipCity;
    }
    public long OrderID { get; set; }
    public string CustomerID { get; set; }
    public int EmployeeID { get; set; }
    public double Freight { get; set; }
    public DateTime OrderDate { get; set; }
    public string ShipCity { get; set; }
}
}

```

Note: You can find the full sample at our [GitHub repository](#).

Perform Grid actions by keyboard shortcut keys

Using keyboard shortcuts, Grid performs navigation and actions.

In addition, You can also perform grid actions with custom keyboard shortcuts. This operation has to be achieved outside of the grid with the help of **keydown** event.

The following example demonstrates on **Adding** a new row when **Enter** key is pressed in the grid.

CSHTML

```
@using Syncfusion.EJ2
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").ValidationRules(new { required = true
}).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

col.Field("Freight").HeaderText("Freight").EditType("numericedit").Format("C
2").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").EditType("dropdownedit").Width("150").Add();
}).AllowPaging().PageSettings(page => page.PageCount(2)).EditSettings(edit
=> {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel" }).Render()
<script>
    var grid = document.getElementById("Grid");
    grid.addEventListener("keydown", keyDownHandler)
    function keyDownHandler(e) {
        if (e.keyCode === 13) {
            var gridIns = grid.ej2_instances[0];
            gridIns.addRecord();
        }
    }
</script>
```

KEYS.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

KEYS.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Customize Pager DropDown

To customize default values of pager dropdown, you need to define in [PageSizes](#) property as array of strings.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowPaging().PageSettings(page => { page.PageSizes((new string[] { "5",
"10", "All" })); }).Render()
```

PAGERDROPDOWN.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Hide the expand/collapse icon in parent row with no record in child grid

By default, the expand/collapse icon will be visible even if the child grid is empty.

You can use [RowDataBound](#) event to hide the icon when there are no records in child grid.

To hide the expand/collapse icon in parent row when no records in child grid, follow the given steps:

Step 1: Create CSS class with custom style to override the default style of Grid.

```
`css
.e-row[aria-selected="true"].e-customizedExpandcell {
background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
background-color: #eee;
}
`
```

Step 2: Add the CSS class to the Grid in the [RowDataBound](#) event handler of Grid.

```
`typescript
function rowDataBound(args) {
var filter = args.data.EmployeeID;
```

```

var data = new ej.data.DataManager(this.childGrid.dataSource).executeLocal(new
ej.data.Query().where("EmployeeID", "equal", parseInt(filter), true));
if (data.length == 0) {
//here hide which parent row has no child records
args.row.querySelector('td').innerHTML = " ";
args.row.querySelector('td').className = "e-customizedExpandcell";
}
}
,

```

In the below demo, the expand/collapse icon in the row with **EmployeeID** as **1** is hidden as it does not have record in child Grid.

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Load("load").RowDataBound("rowDataBound").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").Add();
    col.Field("EmployeeID").HeaderText("Employee ID").Width("150").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().Render()
<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
</style>
<script>
function load(args) {
    var grid = document.getElementById('Grid').ej2_instances[0]
    grid.childGrid = {
        dataSource: dataManger,
        queryString: 'EmployeeID',
        allowPaging: true,
        columns: [
            { field: 'Order', headerText: 'Order ID', textAlign:
'Right', width: 120 },
            { field: 'EmployeeID', headerText: 'Employee ID', width: 120
},
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    }
    var dataManger = [{ Order: 100, ShipName: 'Berlin', EmployeeID: 2 },
        { Order: 101, ShipName: 'Capte', EmployeeID: 3 },
        { Order: 102, ShipName: 'Marlon', EmployeeID: 4 },

```

```

        { Order: 103, ShipName: 'Black pearl', EmployeeID: 5 },
        { Order: 104, ShipName: 'Pearl', EmployeeID: 6 },
        { Order: 105, ShipName: 'Noth bay', EmployeeID: 7 },
        { Order: 106, ShipName: 'baratna', EmployeeID: 8 },
        { Order: 107, ShipName: 'Charge', EmployeeID: 9 },
    ];
    function rowDataBound(args) {
        var filter = args.data.EmployeeID;
        var data = new
ej.data.DataManager(this.childGrid.dataSource).executeLocal(new
ej.data.Query().where("EmployeeID", "equal", parseInt(filter), true));
        if (data.length == 0) {
            //here hide which parent row has no child records
            args.row.querySelector('td').innerHTML = " ";
            args.row.querySelector('td').className = "e-
customizedExpandcell";
        }
    }
}
</script>

```

HIDEARROW.CS

```

public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}

```

Render both EJ1 and EJ2 Grids in same application

To achieve this requirement, you need to use the below code in **_Layout.cshtml** page. Because EJ1 and EJ2 has same library names to perform the different actions. So conflicts may occur when we refer this both controls in same application. To overcome this we need to extend this libraries in ej namespace.

```

`javascript
<script>
var dataCopy = Object.assign({}, ej.data);
$.extend(ej, Syncfusion);
$.extend(ej.data, dataCopy);
</script>
`

```

The Layout page looks like

```

`html
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

@ Syncfusion Essential JS 1 Styles @
@Styles.Render("http://cdn.syncfusion.com/16.3.0.21/js/web/flat-azure/ej.web.all.min.css")
@ Syncfusion Essential JS 2 Styles @
@Styles.Render("https://cdn.syncfusion.com/ej2/styles/compatibility/material.css")
</head>
<body>
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@Scripts.Render("~/Scripts/jsrender.min.js")
@ Syncfusion Essential JS 1 Scripts @
@Scripts.Render("~/Scripts/ej/web/ej.web.all.min.js")
@ Syncfusion Essential JS 2 Scripts @
@Scripts.Render("~/Scripts/ej2/ej2.min.js")
@RenderSection("scripts", required: false)
<script>
var dataCopy = Object.assign({}, ej.data);
$.extend(ej, Syncfusion);
$.extend(ej.data, dataCopy);
</script>
<div class="container body-content">
@RenderBody()
</div>
@Html.EJ().ScriptManager()
@Html.EJS().ScriptManager()
</body>
</html>

```

Set complex column as Foreignkey column

The following example shows how to set the complex column as foreign key column.

In the following example, **Employee.EmployeeID** is a complex column and also declared as a foreign column which shows **FirstName** column from foreign data.

CSHTML

```

@Html.EJS().Grid("ForeignKey").DataSource((IEnumerable<object>) ViewBag.DataSource).Columns(col =>
{

```

```
col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Employee.EmployeeID").ForeignKeyField("EmployeeID").ForeignKeyVal
ue("FirstName").DataSource((IEnumerable<object>)ViewBag.foreign).HeaderText("
Employee Name").Width("150").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
}).Render();
```

FOREIGNKEY.CS

```
public IActionResult DefaultExporting()
{
    var order = OrdersDetails.GetAllRecords();
    ViewBag.DataSource = order;
    var emp = EmployeeView.GetAllRecords();
    ViewBag.foreign = emp;
    return View();
}
```

Complex Data Binding with list of Array Of Objects

The following example shows how to set Complex field for datasource having Array Of Objects.

CSHTML

```
@Html.EJS().Grid("Complex").DataSource((IEnumerable<object>)ViewBag.DataSour
ce).Columns(col =>
{
    col.Field("EmployeeID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAli
gn.Right).Add();
    col.Field("Name.0.FirstName").HeaderText("FirstName").Width("140").Add();

    col.Field("Name.LastName").HeaderText("LastName").Width("130").TextAlign(Syn
cfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("City").Width("120").Add();
    col.Field("Country").Width("140").Add();
}).AllowPaging().Render();
```

COMPLEXARRAY.CS

```
public IActionResult Index()
{
    var ComplexData = ComplexData.GetAllRecords();
    ViewBag.DataSource = ComplexData;
    return View();
}
```

Select grid rows based on certain condition

You can select the specific row in the grid based on a certain condition by using the **selectRows** method in the [DataBound](#) event of Grid.

In the below demo, we have selected the grid rows only when **EmployeeID** column value greater than 3.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("150").Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("120").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().DataBound("dataBound").RowDataBound("rowDataBound").SelectionSettings(select =>
select.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).Render()
<script>
    var selIndex = [];
    function rowDataBound(args) {
        if (args.data['EmployeeID'] > 3) {
            selIndex.push(parseInt(args.row.getAttribute('aria-rowindex')));
        }
    }
    function dataBound(args) {
        if (selIndex.length) {
            this.selectRows(selIndex);
            selIndex = [];
        }
    }
</script>
```

DATABASESELECTION.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Collapse all grouped rows at initial render

You can collapse all the grouped rows at initial rendering by using [DataBound](#) event with **collapseAll** method of the grid.

In the below demo, all the grouped rows are collapsed at initial rendering.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.dataSource)
.DataBound("bound").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
```

```

col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width("150").EditType("dropdowndedit").Add();
}).AllowPaging().AllowGrouping().GroupSettings(group => group.Columns(new string[] { "ShipCountry" })).Render()
<script>
    var initial = true;
    function bound() {
        if (initial === true) {
            this.groupModule.collapseAll();
            initial = false;
        }
    }
</script>

```

COLLAPSEALL.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}

```

How to show grouped rows based on the pageSize

By default, we have displayed the no of records based on the [PageSize](#). If you want to show grouped column rows based on the [PageSize](#) then we suggest you to use the below way.

In the below sample, we have overridden the default **generateQuery** to display the grouped rows instead of grid rows based on the [PageSize](#).

CSHTML

```

<script>
    var old = ej.grids.Data.prototype.generateQuery;
    ej.grids.Data.prototype.generateQuery = function () {
        var query = old.call(this, true);
        this.pageQuery(query);
        return query;
    };
</script>
@Html.EJS().Grid("FlatGrid").DataSource((IEnumerable<object>)ViewBag.DataSource).AllowSorting().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}

```



```
}).AllowPaging().PageSettings(page =>
page.PageSize(5)).AllowGrouping().GroupSettings(group=>group.Columns(new
string[] { "ShipCountry" })).Render()
```

GROUP-PAGE-SIZE.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Get specific row and cell index in Grid

You can get the specific row and cell index of the grid by using [RowSelected](#) event of the grid. Here, we can get the row and cell index by using **aria-rowindex**(get row Index from **tr** element) and **aria-colindex**(column index from **td** element) attribute.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.RowSelected("rowSelected").Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipName").HeaderText("Ship Name").Width("150").Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").EditType("dropdownedit").Add();
}).AllowPaging().Render()
<script>
function rowSelected(args) {
    alert("row index: " + args.row.getAttribute('aria-rowindex'));
    alert("column index: " + args.target.getAttribute('aria-colindex'));
}
</script>
```

ROWCELLIDX.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

Display the null date values at bottom of the Grid while perform sorting

By default the null values are displayed at bottom of the Grid row while perform sorting in ascending order. As well as this values are displayed at top of the Grid row while perform sorting with descending

order. But you can customize this default order to display the null values at always bottom row of the Grid by using [SortComparer](#) property of [Column](#).

In the below code we have displayed the null date values at bottom of the Grid row while sorting the **OrderDate** column in both ways.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.AllowSorting().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").Width("100").Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width("120").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Format("yMd").SortComparer("sortComparer").Width("130").Add();
    col.Field("ShipCountry").HeaderText("ShipCountry").Width("150").Add();
}).ActionBegin("actionBegin").Render()
<script type="text/javascript">
    var action;
    function actionBegin(args) {
        if (args.requestType == "sorting") {
            action = args.direction;
        }
    }
    function sortComparer(reference, comparer) {
        var sortAsc = action === "Ascending" ? true : false;
        if (sortAsc && reference === null) {
            return 1;
        }
        else if (sortAsc && comparer === null) {
            return -1;
        }
        else if (!sortAsc && reference === null) {
            return -1;
        }
        else if (!sortAsc && comparer === null) {
            return 1;
        }
        else {
            return reference - comparer;
        }
    }
</script>
```

SORT-COMPARER.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Avoid TypeScript Compilation

Syncfusion.EJ2.Javascript includes typescript declaration files. If your application is not configured to compile typescript then exception may occur. To resolve this we need to prevent the typescript compilation during MS build process by adding the below line in **.csproj** file.

```
`cs
<PropertyGroup>
<TypeScriptCompileBlocked>True</TypeScriptCompileBlocked>
</PropertyGroup>
`
```

Enable editing in single click

Normal Editing

You can make a row editable on a single click with **Normal** mode of editing in Grid, by using the **startEdit** and **endEdit** methods.

Bind the **mouseup** event for Grid and in the event handler call the **startEdit** and **endEdit**, based on the clicked target element.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>) ViewBag.DataSource)
.AllowPaging().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(100).IsPrimaryKey
(true).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width(120).Add();

col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.Te
xtAlign.Right).Width(120).Format("C2").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width(150).Add();
}).Load("load").EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel" }).Render()
<script>
function load(args) {
    var instance = document.getElementById('Grid').ej2_instances[0];
    instance.element.addEventListener('mouseup', function (e) {
        if (e.target.classList.contains("e-rowcell")) {
            if (instance.isEdit)
                instance.endEdit();
            var index = parseInt(e.target.getAttribute("Index"));
            instance.selectRow(index);
            instance.startEdit();
        }
    });
}
</script>
```

SINGLE-CLICK-NORMAL-EDIT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Open dropdown edit popup on single click

You can open the default dropdown edit popup with single click edit by focusing the dropdown element and calling the EJ2 dropdown list's **showPopup** method in the Grid's **ActionComplete** event. In this demo we have used a global flag variable in the **mouseup** event to ensure if the dropdown column is the clicked edit target.

CSHTML

```
@Html.EJS().Grid("Grid").ActionComplete("onActionComplete").DataSource((IEnumerable<object>)ViewBag.DataSource).AllowPaging().Columns(col =>
{
    col.Field("OrderID").HeaderText("Order ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(100).IsPrimaryKey(true).Add();
    col.Field("CustomerID").HeaderText("Customer ID").Width(120).Add();

    col.Field("Freight").HeaderText("Freight").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(120).Format("C2").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").EditType("dropdownedit").Width(150).Add();
}).Load("load").EditSettings(edit => {
    edit.AllowAdding(true).AllowDeleting(true).Mode(Syncfusion.EJ2.Grids.EditMode.Normal);
}).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel" }).Render()
<script>
    var isDropdown = false;

    function load(args) {
        var instance = document.getElementById('Grid').ej2_instances[0];
        instance.element.addEventListener('mouseup', function (e) {
            if (e.target.classList.contains("e-rowcell")) {
                if (instance.isEdit)
                    instance.endEdit();
                var rowInfo = instance.getRowInfo(e.target);
                if (rowInfo.column.field === "ShipCountry")
                    isDropdown = true;
                instance.selectRow(rowInfo.rowIndex);
                instance.startEdit();
            }
        });
    }

    function onActionComplete(args) {
        if (args.requestType === "beginEdit" && isDropdown) {
            isDropdown = false;
            var dropdownObj = args.form.querySelector('.e-dropdownlist').ej2_instances[0];
            dropdownObj.element.focus();
            dropdownObj.showPopup();
        }
    }
}
```

```
}
</script>
```

OPEN-DROPDOWN-POPUP.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Cascading DropDownList with Grid editing

You can achieve the Cascading DropDownList with grid Editing by using the Cell Edit Template feature.

In the below demo, Cascading DropDownList is rendered for the **ShipCountry** and **ShipState** column.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.dataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign
n.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Width("150").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").EditType("dropdownedit").Edit(new
{create="countryCreate", read="countryRead", destroy="countryDestroy",
write="countryWrite" }).Add();
    col.Field("ShipState").HeaderText("Ship
State").Width("150").EditType("dropdownedit").Edit(new
{create="stateCreate", read="stateRead", destroy="stateDestroy",
write="stateWrite" }).Add();
}).AllowPaging().EditSettings(edit => {
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.Grids.EditMode.Normal); }).Toolbar(new List<string>() { "Add", "Edit",
"Delete", "Update", "Cancel" }).Render()
<script>
var countryElem;
var countryObj;
var stateElem;
var stateObj;
var country = [
    { countryName: 'United States', countryId: '1' },
    { countryName: 'Australia', countryId: '2' }
];
var state = [
    { stateName: 'New York', countryId: '1', stateId: '101' },
    { stateName: 'Virginia', countryId: '1', stateId: '102' },
    { stateName: 'Washington', countryId: '1', stateId: '103' },
    { stateName: 'Queensland', countryId: '2', stateId: '104' },
```

```

        { stateName: 'Tasmania', countryId: '2', stateId: '105' },
        { stateName: 'Victoria', countryId: '2', stateId: '106' }
    ];
    function countryCreate() {
        countryElem = document.createElement('input');
        return countryElem;
    }
    function countryRead() {
        return countryObj.text;
    }
    function countryDestroy() {
        countryObj.destroy();
    }
    function countryWrite() {
        countryObj = new ej.dropdowns.DropDownList({
            dataSource: country,
            fields: { value: 'countryId', text: 'countryName' },
            change: function() {
                stateObj.enabled = true;
                var tempQuery = new
ej.data.Query().where('countryId', 'equal', countryObj.value);
                stateObj.query = tempQuery;
                stateObj.text = null;
                stateObj.dataBind();
            },
            placeholder: 'Select a country',
            floatLabelType: 'Never'
        });
        countryObj.appendTo(countryElem);
    }
    function stateCreate() {
        stateElem = document.createElement('input');
        return stateElem;
    }
    function stateRead() {
        return stateObj.text;
    }
    function stateDestroy() {
        stateObj.destroy();
    }
    function stateWrite() {
        stateObj = new ej.dropdowns.DropDownList({
            dataSource: state,
            fields: { value: 'stateId', text: 'stateName' },
            enabled: false,
            placeholder: 'Select a state',
            floatLabelType: 'Never'
        });
        stateObj.appendTo(stateElem);
    }
}
</script>

```

CASCADING.CS

```

public IActionResult Index()
{

```

```

ViewBag.DataSource = OrderDetails.GetAllRecords();
return View();
}

```

Hide sorting options on excel filter Dialog

You can hide the sorting options on the excel filter dialog by setting display as none for the following classes.

```

`css
.e-excel-ascending,
.e-excel-descending,
.e-separator.e-excel-separator {
display: none;
}
`

```

CSHTML

```

@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
    col.Field("OrderDate").HeaderText("Order
Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("
yMd").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
}).AllowFiltering().AllowPaging().FilterSettings(filter => {
filter.Type(Syncfusion.EJ2.Grids.FilterType.Excel); }).Render()
<style>
    .e-excel-ascending,
    .e-excel-descending,
    .e-separator.e-excel-separator {
        display: none;
    }
</style>

```

DISABLE-SORTING-EXCEL.CS

```

public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}

```

Add a title to the header when using Grid print action

You can add your title in the header through an [BeforePrint](#) event. With the help of this event you can add your title element as you want.

CSHTML

```
@Html.EJS().Grid("Grid").DataSource((IEnumerable<object>)ViewBag.DataSource)
.Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width("120").Add();
    col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();

    col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("ShipCity").HeaderText("Ship City").Width("120").Add();
    col.Field("ShipCountry").HeaderText("Ship Country").Width("150").Add();
}).AllowPaging().BeforePrint("beforePrint").Toolbar(new List<string>() {
"Print" }).Render()
<script>
    function beforePrint(args) {
        var div = document.createElement("Div")
        div.innerHTML = "Title here"
        div.style.textAlign = 'center';
        div.style.color = 'red';
        div.style.padding = '10px 0';
        div.style.fontSize = '25px';
        args.element.insertBefore(div, args.element.childNodes[0]);
    }
</script>
```

TITLEPRINT.CS

```
public IActionResult Index()
{
    var Order = OrderDetails.GetAllRecords();
    ViewBag.DataSource = Order;
    return View();
}
```

Customizing Filter Dialog by using an additional Parameter

You can customize the default settings of the components which are used in Menu filter by using params of filter property in column definition. In the below sample, OrderID and Freight Columns are numeric columns, while open the filter dialog then you can see that NumericTextBox with spin button is displayed to change/set the filter value. Now using the params option we hide the spin button in NumericTextBox for OrderID Column.

CSHTML

```
@Html.EJS().Grid("ExcelFilter").DataSource((IEnumerable<object>)ViewBag.Data
Source).AllowFiltering().FilterSettings(Filter =>
Filter.Type(Syncfusion.EJ2.Grids.FilterType.Menu)).Columns(col =>
{
```



```

        col.Field("OrderID").HeaderText("Order ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Filter((new { @params = new { showSpinButton = false } })).Add();
        col.Field("CustomerID").HeaderText("Customer Name").Width("150").Add();
        col.Field("OrderDate").HeaderText("Order Date").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Add();

col.Field("Freight").HeaderText("Freight").Width("120").Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
        col.Field("ShipCountry").HeaderText("Ship Country").Width("120").Add();
    }).AllowPaging().Render()

```

EXCELFILTER.CS

```

public IActionResult Index()
{
    if (orders.Count() == 0)
        DataSource();
    ViewBag.Datasource = orders;
    return View();
}

public void DataSource()
{
    int code = 10000;
    for (int i = 1; i < 10; i++)
    {
        orders.Add(new OrderDetails(code + 1, "ALFKI", i + 0, 2.3 * i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro", "Denmark", new DateTime(1996, 7, 16), "Kirchgasse 6"));
        orders.Add(new OrderDetails(code + 2, "ANATR", i + 2, 3.3 * i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha", "Brazil", new DateTime(1996, 9, 11), "Avda. Azteca 123"));
        orders.Add(new OrderDetails(code + 3, "ANTON", i + 1, 4.3 * i, true, new DateTime(1957, 11, 30), "Cholchester", "Frankenversand", "Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave. Bolívar #65-98 Llano Largo"));
        orders.Add(new OrderDetails(code + 4, "BLONP", i + 3, 5.3 * i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel", "Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
        orders.Add(new OrderDetails(code + 5, "BOLID", i + 4, 6.3 * i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes", "Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
        code += 5;
    }
}

```

Customize the Empty Record Template

The empty record template feature in the Grid allows you to use custom content such as images, text, or other components, when the grid doesn't contain any records to display. This feature replaces the default message of 'No records to display' typically shown in the grid.

To activate this feature, set the `emptyRecordTemplate` property of the Grid. The `emptyRecordTemplate` property expects the HTML element or a function that returns the HTML element.

In the following example, an image and text have been rendered as a template to indicate that the grid has no data to display.

CSHTML

```
var DropDownList = new Syncfusion.EJ2.DropDowns.DropDownList() { DataSource
= ViewBag.datasource, Fields = new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings() { Value =
"ShipCountry", Text = "ShipCountry" } };
@Html.EJS().Grid("EmptyRecordTemplate").EmptyRecordTemplate("#emptytemplate"
).Columns(col =>
{
    col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width("140").ValidationRules(new { required = true,
number = true }).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("CustomerID").HeaderText("Customer
Name").Width("150").ValidationRules(new { required = true, minLength = 3
}).Add();

col.Field("Freight").HeaderText("Freight").Width("140").EditType("numericedit").ValidationRules(new { required = true
}).Format("C2").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
    col.Field("OrderDate").HeaderText("Order
Date").EditType("datetimepickeredit").Width("150").Format(new { type =
"datetime", format = "M/d/y hh:mm a" }).Add();
    col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").EditType("dropdownedit").Edit(new { @params =
DropDownList }).Add();

col.Field("Verified").HeaderText("Verified").EditType("booleanedit").Display
AsCheckBox(true).Type("boolean").Width("150").Add();
}).AllowPaging().PageSettings(page => page.PageCount(5)).EditSettings(edit
=> { edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true);
}).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update", "Cancel"
}).Render()
<style type="text/css" class="cssStyles">
.emptyRecordTemplate {
    text-align: center;
}
.e-emptyRecord {
    display: block;
    margin: 10px auto;
}
</style>
<script id="emptytemplate" type="text/x-template">
    <div class='emptyRecordTemplate'>
        
        <span>There is no data available to display at the moment.</span>
    </div>
</script>
```

EMPTY-RECORD-TEMPLATE.CS

```
public IActionResult Index()
{
    ViewBag.DataSource = OrderDetails.GetAllRecords();
    return View();
}
```

HeatMap Chart

Getting Started with ASP.NET MVC HeatMapChart Control

This section briefly explains about how to include [ASP.NET MVC HeatMapChart](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

<add namespace="Syncfusion.EJ2"/>

</namespaces>

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC HeatMapChart control

Now, add the Syncfusion ASP.NET MVC HeatMapChart control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().HeatMap("container").Render()
```

Populate heat map with data

This section explains how to populate the following two-dimensional array data to the heat map.

CSHTML

```
@model int[,]
@Html.EJS().HeatMap("container").DataSource(Model).Render()
```

HOMECONTROLLER.CS

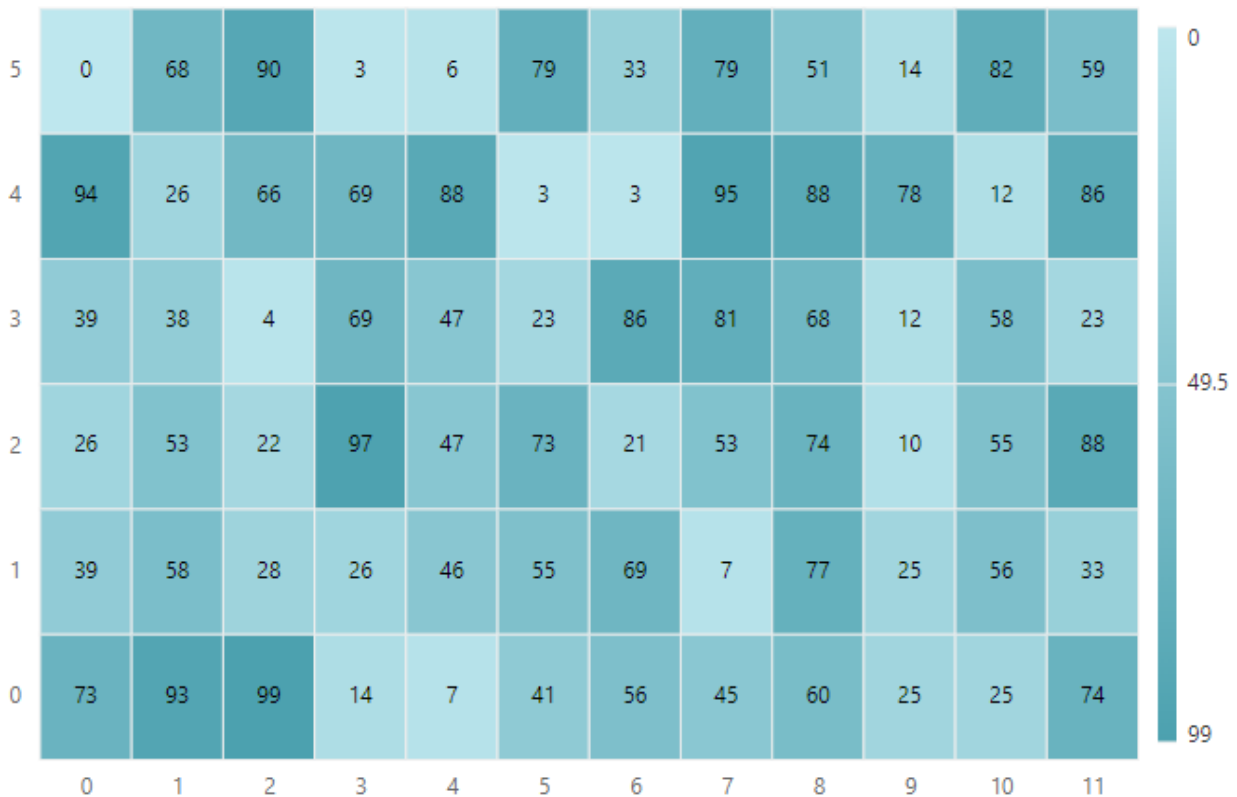
```
public ActionResult Index()
{
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet", "Margaret",
    "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin", "Mario" };
}
```

```

ViewBag.xLabels = xlabels;
string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
"Sat" };
ViewBag.yLabels = yLabels;
int[,] data = new int[,]
{
{73, 39, 26, 39, 94, 0},
{93, 58, 53, 38, 26, 68},
{99, 28, 22, 4, 66, 90},
{14, 26, 97, 69, 69, 3},
{7, 46, 47, 47, 88, 6},
{41, 55, 73, 23, 3, 79},
{56, 69, 21, 86, 3, 33},
{45, 7, 53, 81, 95, 79},
{60, 77, 74, 68, 88, 51},
{25, 25, 10, 12, 78, 14},
{25, 56, 55, 58, 12, 82},
{74, 33, 88, 23, 86, 59}
};
return View(data);
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC HeatMapChart control will be rendered in the default web browser.



Enable axis labels

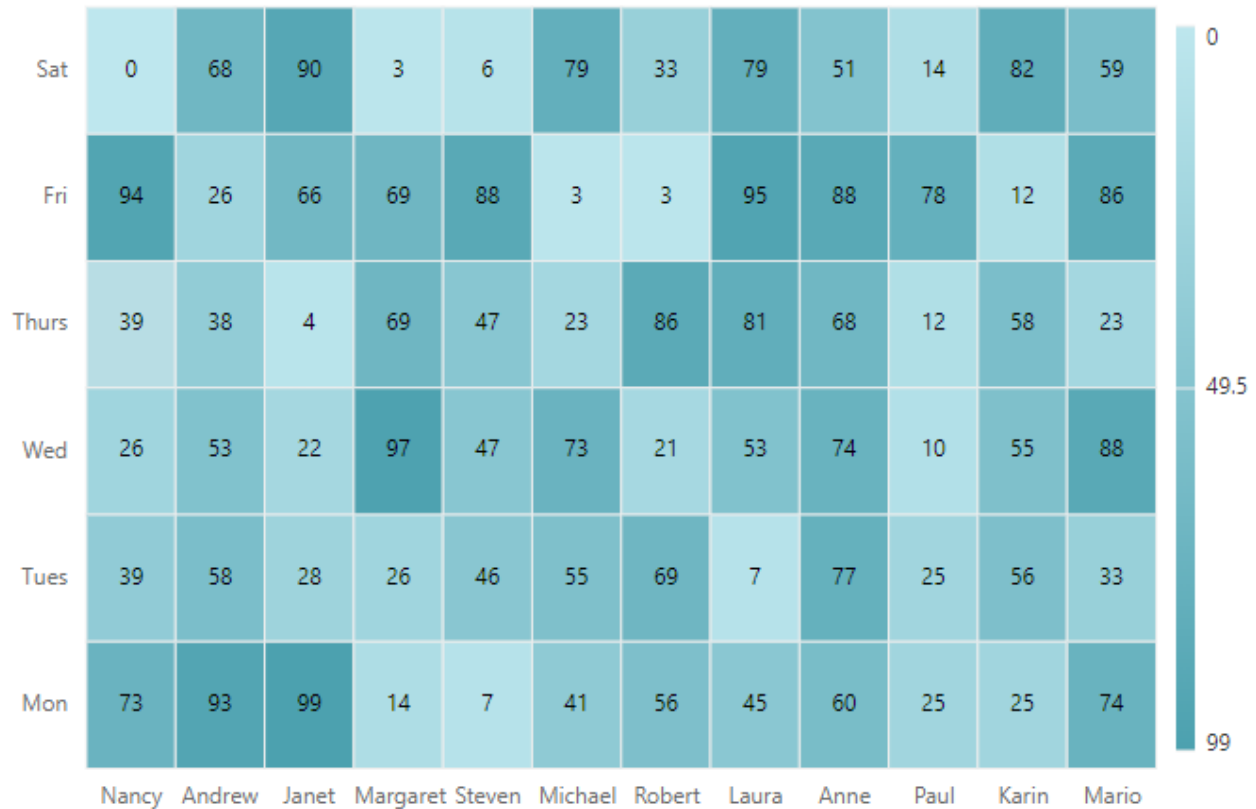
You can add axis labels to the heat map and format those labels using the [xAxis](#) and [yAxis](#) properties. Axis labels provide additional information about the data points populated in the heat map.

CSHTML

```
@model int[,]
@Html.EJS().HeatMap("container").XAxis(
    xAxis =>
    {
        xAxis.Labels(ViewBag.xLabels);
    }).YAxis(yaxis =>
    {
        yaxis.Labels(ViewBag.yLabels);
    }).DataSource(Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet", "Margaret",
    "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin", "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    int[,] data = new int[, ]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return View(data);
}
```



Add heat map title

Add a title using the [TitleSettings](#) property to the heat map to provide quick information to the user about the data populated in the heat map.

CSHTML

```
@model int[,]
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Sales Revenue
per Employee (in 1000 US$)").TextStyle(ViewBag.textStyle)).XAxis(
    xAxis =>
    {
        xAxis.Labels(ViewBag.xLabels);
    }).YAxis(yaxis =>
    {
        yaxis.Labels(ViewBag.yLabels);
    }).DataSource(Model).Render()
```

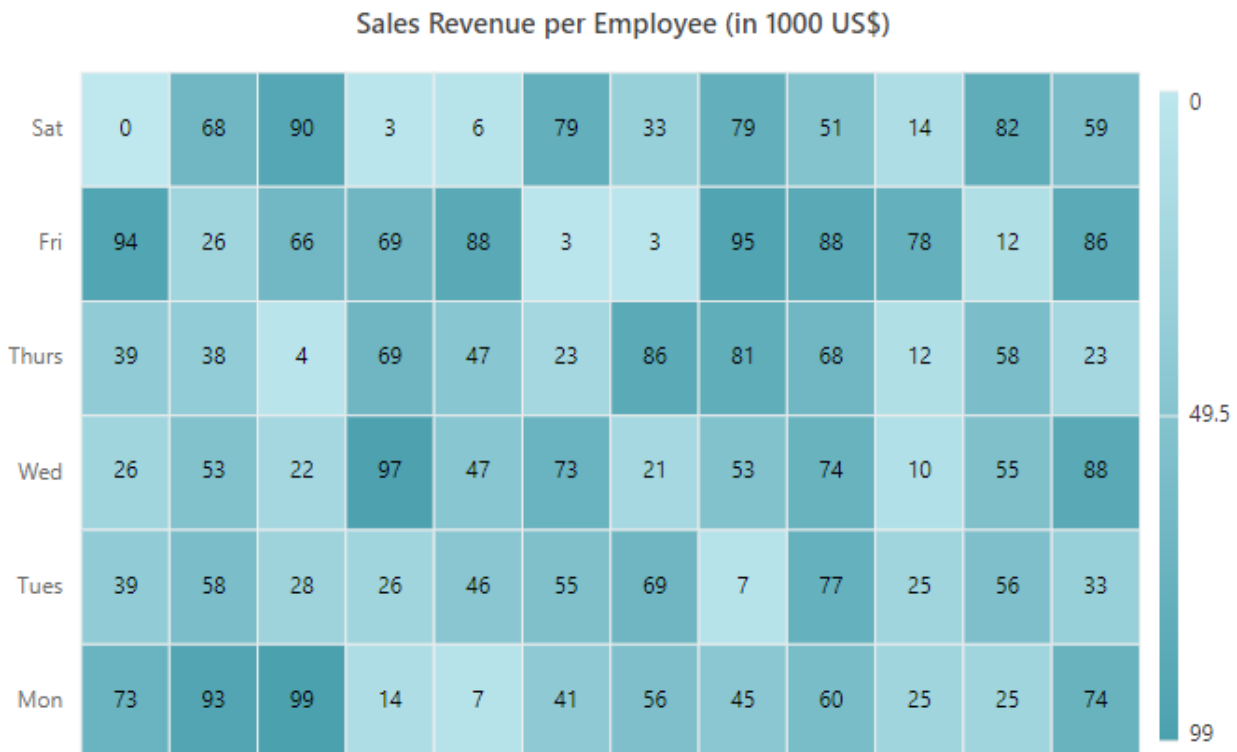
HOMECONTROLLER.CS

```
public ActionResult Index()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
```

```

};
string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet", "Margaret",
"Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin", "Mario" };
ViewBag.xLabels = xlabels;
string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
"Sat" };
ViewBag.yLabels = yLabels;
int[,] data = new int[,]
{
    { 73, 39, 26, 39, 94, 0 },
    { 93, 58, 53, 38, 26, 68 },
    { 99, 28, 22, 4, 66, 90 },
    { 14, 26, 97, 69, 69, 3 },
    { 7, 46, 47, 47, 88, 6 },
    { 41, 55, 73, 23, 3, 79 },
    { 56, 69, 21, 86, 3, 33 },
    { 45, 7, 53, 81, 95, 79 },
    { 60, 77, 74, 68, 88, 51 },
    { 25, 25, 10, 12, 78, 14 },
    { 25, 56, 55, 58, 12, 82 },
    { 74, 33, 88, 23, 86, 59 }
};
return View(data);
}

```

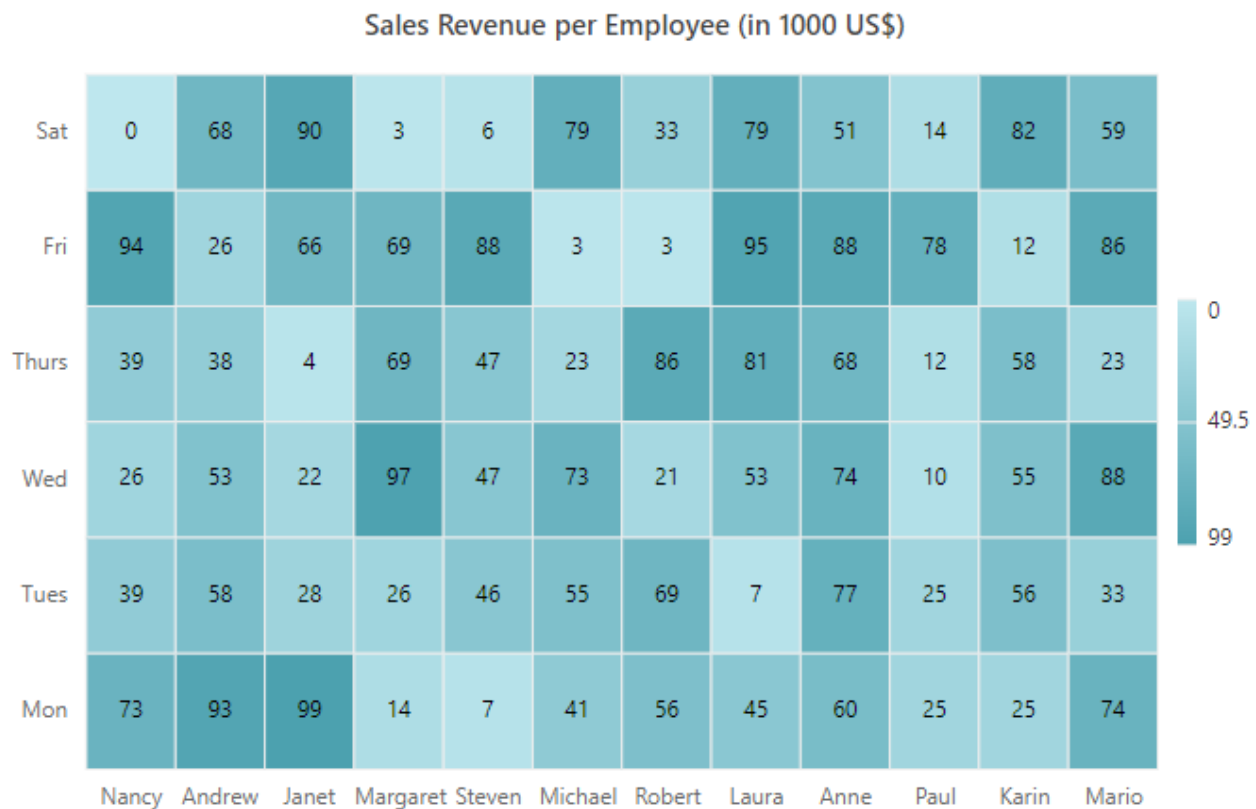


Enable legend

Use a legend for the heat map in the [LegendSettings](#) object by setting the [Visible](#) property to **true**.

CSHTML


```
@model int[,]
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Sales Revenue
per Employee (in 1000 US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.Labels(ViewBag.xLabels);
}).YAxis(yaxis =>
{
    yaxis.Labels(ViewBag.yLabels);
}).LegendSettings(ls
=>ls.Visible(true).Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right).Sho
wLabel(true).Height("150px")).DataSource(Model).Render()
```



Add data label

Add data labels to improve the readability of the heat map. This can be achieved by setting the [ShowLabel](#) property to **true** in the [CellSettings](#) object.

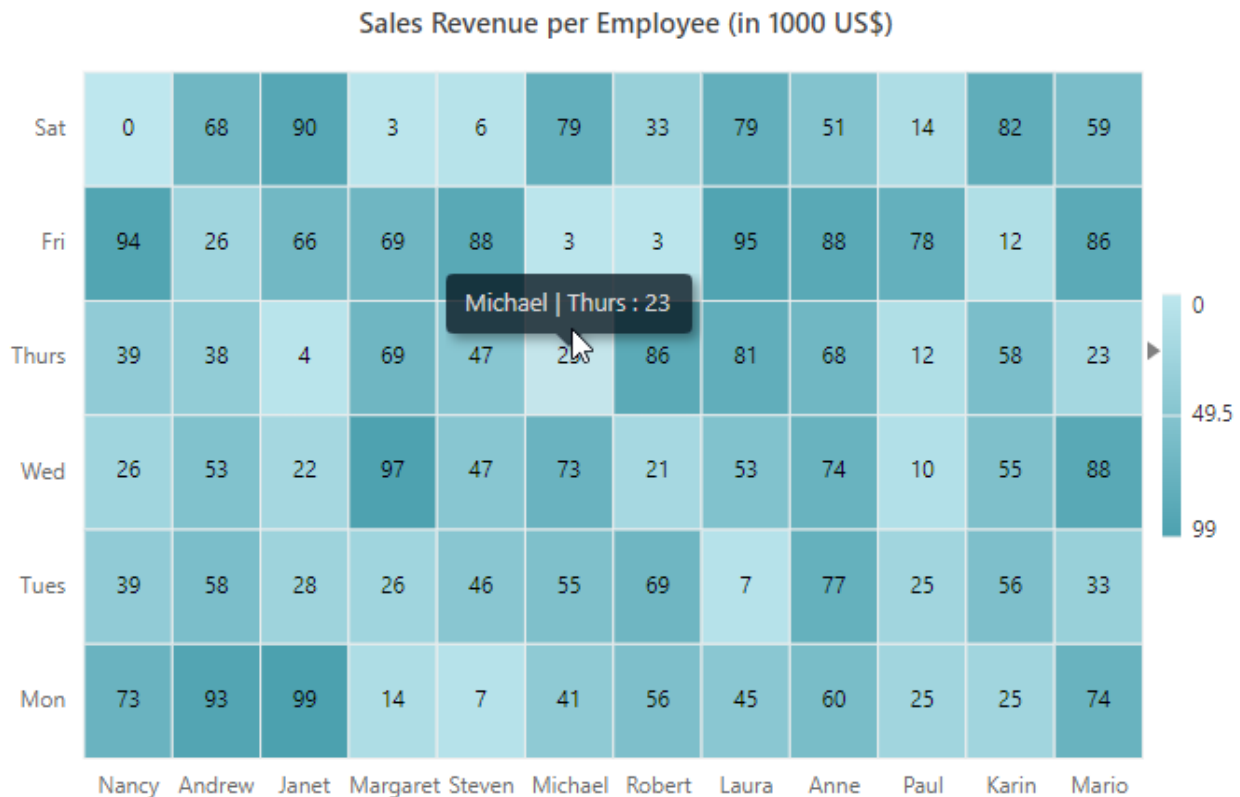
CSHTML

```
@model int[,]
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Sales Revenue
per Employee (in 1000 US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.Labels(ViewBag.xLabels);
}).YAxis(yaxis =>
{
    yaxis.Labels(ViewBag.yLabels);
```

```

    }).LegendSettings(ls
=>ls.Visible(true).Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right).ShowLabel(true).Height("150px")).CellSettings(cs
=>cs.ShowLabel(true)).DataSource(Model).Render()

```



Add custom cell palette

The default palette settings of the heat map cells can be customized by using the [PaletteSettings](#) property. Using the [Palette](#) property in [paletteSettings](#) object, you can change the color set for the cells. You can change the color mode of the cells to fixed or gradient mode using the [Type](#) property.

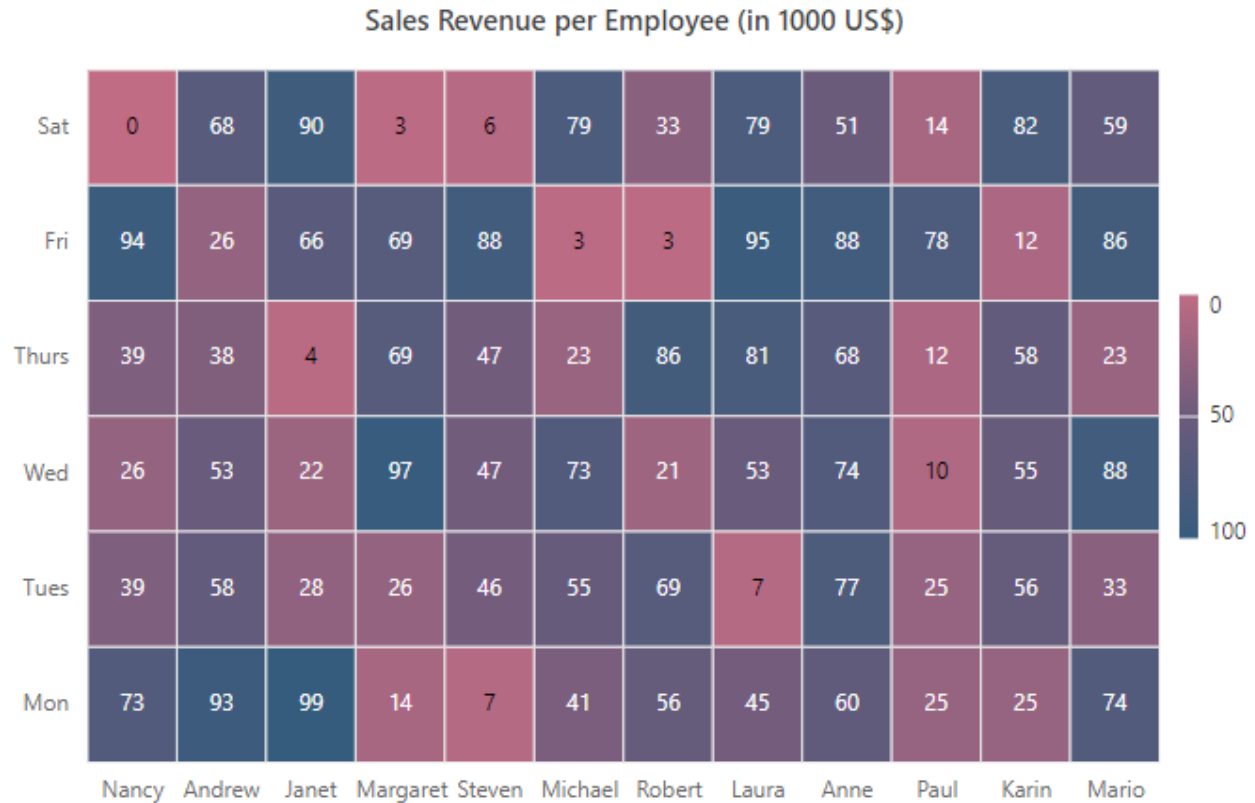
CSHTML

```

@model int[,]
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Sales Revenue
per Employee (in 1000 US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.Labels(ViewBag.xLabels);
}).YAxis(yaxis =>
{
    yaxis.Labels(ViewBag.yLabels);
}).LegendSettings(ls
=>ls.Visible(true).Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right).ShowLabel(true).Height("150px")).CellSettings(cs
=>cs.ShowLabel(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Value(0).Add();
    palette.Color("#6C5B7B").Value(50).Add();
}

```

```
palette.Color("#355C7D").Value(100).Add();
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient).DataSource(Model).Render();
```



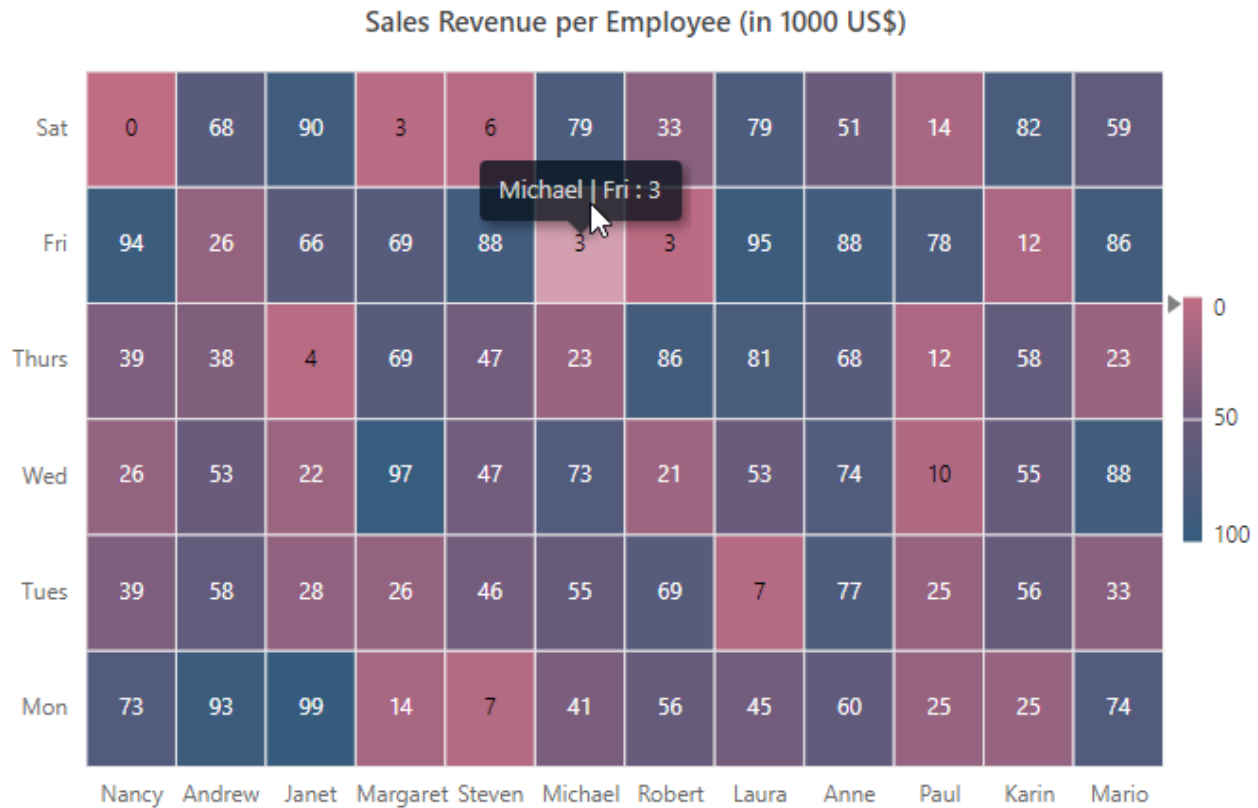
Enable tooltip

The tooltip is used when you cannot display information by using the data labels due to space constraints. You can enable the tooltip by setting the [ShowTooltip](#) property to **true**.

CSHTML

```
@model int[,]
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Sales Revenue
per Employee (in 1000 US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.Labels(ViewBag.xLabels);
}).YAxis(yaxis =>
{
    yaxis.Labels(ViewBag.yLabels);
}).LegendSettings(ls
=>ls.Visible(true).Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right).Sho
wLabel(true).Height("150px")).CellSettings(cs
=>cs.ShowLabel(true).ShowTooltip(true).PaletteSettings(ps =>
ps.Palette(palette =>
{
    palette.Color("#C06C84").Value(0).Add();
    palette.Color("#6C5B7B").Value(50).Add();
```

```
palette.Color("#355C7D").Value(100).Add();
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient).DataSource(Model).Render();
```



Note: [View Sample in GitHub.](#)

Note: You can explore our [ASP.NET MVC HeatMap Chart example](#) that shows you how to render the HeatMap Chart in ASP.NET MVC.

Working with data in ASP.NET MVC HeatMap Chart Component

Heat map visualizes the JSON data and two-dimensional array data. Using the data adaptor support, data can be bound to the heat map.

Data adaptor

Heat map supports the following types of data binding with the adaptor support.

- Array
- Table Binding
- Cell Binding
- JSON data
- Table Binding
- Cell Binding

Array - table binding

This data type is a collection of one dimensional array objects, at which each inner array contains data points for an X-axis data label. This is the default data binding type for heat map. You can also directly bind the array object to the [dataSource](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("GDP Growth Rate for Major Economies (in
Percentage)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1,
1)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years).LabelFormat("yyy
y");
}).YAxis(yaxis =>
yaxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.Numeric)).LegendSettings(ls
=>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()
```

ARRAY-ROW.CS

```
public ActionResult ArrayRow()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "China", "India", "Australia",
    "Mexico", "Canada", "Brazil",
    "USA", "UK", "Germany", "Russia", "France", "Japan"};
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[10] { "2008", "2009", "2010", "2011",
    "2012", "2013", "2014", "2015", "2016", "2017" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private double[,] GetDataSource()
{
    double[,] data = new double[12, 10]
    {
        { 9.5, 2.2, 4.2, 8.2, -0.5, 3.2, 5.4, 7.4, 6.2, 1.4 },
        { 4.3, 8.9, 10.8, 6.5, 5.1, 6.2, 7.6, 7.5, 6.1, 7.6 },
        { 3.9, 2.7, 2.5, 3.7, 2.6, 5.1, 5.8, 2.9, 4.5, 5.1 },
        { 2.4, -3.7, 4.1, 6.0, 5.0, 2.4, 3.3, 4.6, 4.3, 2.7 },
        { 2.0, 7.0, -4.1, 8.9, 2.7, 5.9, 5.6, 1.9, -1.7, 2.9 },
        { 5.4, 1.1, 6.9, 4.5, 2.9, 3.4, 1.5, -2.8, -4.6, 1.2 },
        { -1.3, 3.9, 3.5, 6.6, 5.2, 7.7, 1.4, -3.6, 6.6, 4.3 },
        { -1.6, 2.3, 2.9, -2.5, 1.3, 4.9, 10.1, 3.2, 4.8, 2.0 },
        { 10.8, -1.6, 4.0, 6.0, 7.7, 2.6, 5.6, -2.5, 3.8, -1.9 },
        { 6.2, 9.8, -1.5, 2.0, -1.5, 4.3, 6.7, 3.8, -1.2, 2.4 },
    }
```

```

                {1.2, 10.9, 4.0, -1.4, 2.2, 1.6, -2.6, 2.3, 1.7, 2.4},
                {5.1, -2.4, 8.2, -1.1, 3.5, 6.0, -1.3, 7.2, 9.0, 4.2}
            };
            return data;
        }
    }

```

Array - cell binding

This data type is a collection of array objects that contain information about the row index, column index, and data value for each cell. You can bind the data to heat map by using the `data` property in the [dataSource](#) and setting the `adaptorType` property to `Cell`.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Percentage of
Individuals Using Internet by
Country").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).PaletteSetting
s(ps => ps.Palette(palette =>
{
    palette.Color("#3498DB").Add();
    palette.Color("#2C3E50").Add();
}))).CellSettings(cs
=>cs.Border(ViewBag.border).TextStyle(ViewBag.cellTextStyle).Format("{value}
%")).LegendSettings(ls =>
ls.Visible(false)).TooltipRender("tooltipRender").DataSourceSettings(ds => {
ds.IsJsonData(false).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell);
}).Render()
<script>
    var tooltipRender = function (args) {
        args.content = [args.yLabel + ' | ' + args.xLabel + ' : ' +
args.value + ' %'];
    };
</script>

```

ARRAY-CELL.CS

```

public ActionResult ArrayCell()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "China", "India", "Australia",
    "Mexico", "Canada", "Brazil",
    "USA", "UK", "Germany", "Russia", "France", "Japan" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[7] { "2000", "2005", "2010", "2011",
    "2012", "2013", "2014" };
    ViewBag.yLabels = yLabels;
    ViewBag.border = new { width = 0 };
}

```

```

        ViewBag.cellTextStyle = new { color = "white" };
        double[,] cellData = new double[,]
        {
            {0, 0, 10.75}, {0, 1, 14.5}, {0, 2, 25.5}, {0, 3, 39.5}, {0,
4, 59.75}, {0, 5, 35.50}, {0, 6, 75.5},
            {1, 0, 20.75}, {1, 1, 35.5}, {1, 2, 29.5}, {1, 3, 75.5}, {1,
4, 80}, {1, 5, 65}, {1, 6, 85},
            {2, 0, 6}, {2, 1, 18.5}, {2, 2, 30.05}, {2, 3, 35.5}, {2, 4,
40.75}, {2, 5, 50.75}, {2, 6, 65},
            {3, 0, 30.5}, {3, 1, 20.5}, {3, 2, 45.30}, {3, 3, 50}, {3,
4, 55}, {3, 5, 85.80}, {3, 6, 87.5},
            {4, 0, 10.5}, {4, 1, 20.75}, {4, 2, 35.5}, {4, 3, 35.5}, {4,
4, 45.5}, {4, 5, 65}, {4, 6, 75.5},
            {5, 0, 45.5}, {5, 1, 20.75}, {5, 2, 45.5}, {5, 3, 50.75},
{5, 4, 79.30}, {5, 5, 84.20}, {5, 6, 87.36},
            {6, 0, 26.82}, {6, 1, 70}, {6, 2, 75}, {6, 3, 79.5}, {6, 4,
88.5}, {6, 5, 89.5}, {6, 6, 91.75},
            {7, 0, 15.75}, {7, 1, 20.75}, {7, 2, 25.5}, {7, 3, 42.35},
{7, 4, 45.15}, {7, 5, 76.5}, {7, 6, 80.5},
            {8, 0, 1.98}, {8, 1, 15.23}, {8, 2, 43}, {8, 3, 49}, {8, 4,
63.80}, {8, 5, 67.97}, {8, 6, 70.52},
            {9, 0, 14.31}, {9, 1, 42.87}, {9, 2, 77.28}, {9, 3, 77.82},
{9, 4, 81.44}, {9, 5, 81.92}, {9, 6, 83.75},
            {10, 0, 25.5}, {10, 1, 35.5}, {10, 2, 40.5}, {10, 3, 45.05},
{10, 4, 50.5}, {10, 5, 75.5}, {10, 6, 90.58}
        };

        ViewBag.dataSource = cellData;
        return View();
    }

```

JSON data - table binding

In JSON table data binding, each JSON object contains an X-axis data point as row header and all the corresponding Y-axis data values. You can bind the JSON table data to the heat map using the `data` property in `dataSource`. To achieve this, you should enable the `isJsonData` property and define the `adaptorType` property as **Table**. The `xDataMapping` property is used to map the row header in JSON data.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Olympic Medal
Achievements of most Successful
Countries").TextStyle(ViewBag.textStyle)).XAxis(
xAxis =>
{
    xAxis.LabelRotation(45)
    .LabelIntersectAction(Syncfusion.EJ2.HeatMap.LabelIntersectAction.None)
    .Labels(ViewBag.xLabels);
}).YAxis(yaxis =>
{
    yaxis.Title(ViewBag.title)
    .Labels(ViewBag.yLabels);
}).PaletteSettings(ps => ps.Palette(palette =>
{

```

```

        palette.Color("#F0C27B").Add();
        palette.Color("#4B1248").Add();
    })).LegendSettings(ls => ls.Visible(false)).CellSettings(cs =>
cs.Border(ViewBag.border)).DataSource("dataSource").TooltipRender("tooltipRe
nder").
DataSourceSettings(ds => {
ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Table).XD
ataMapping("Region"); }).Render()
<script>
    var tooltipRender = function (args) {
        args.content = [args.yLabel + ' | ' + args.xLabel + ' : ' +
args.value + ' medals'];
    };
    var dataSource = [
        { "Region": "USA", "2000": 93, "2004": 101, "2008": 112, "2012":
103, "2016": 121 },
        { "Region": "GBR", "2000": 28, "2004": 30, "2008": 49, "2012":
65, "2016": 67 },
        { "Region": "China", "2000": 58, "2004": 63, "2008": 100,
"2012": 91, "2016": 70 },
        { "Region": "Russia", "2000": 89, "2004": 90, "2008": 60,
"2012": 69, "2016": 55 },
        { "Region": "Germany", "2000": 56, "2004": 49, "2008": 41,
"2012": 44, "2016": 42 },
        { "Region": "Japan", "2000": 18, "2004": 37, "2008": 25, "2012":
38, "2016": 41 },
        { "Region": "France", "2000": 38, "2004": 33, "2008": 43,
"2012": 35, "2016": 42 },
        { "Region": "KOR", "2000": 28, "2004": 30, "2008": 32, "2012":
30, "2016": 21 },
        { "Region": "Italy", "2000": 34, "2004": 32, "2008": 27, "2012":
28, "2016": 28 },
    ];
</script>

```

JSON-ROW.CS

```

public ActionResult JsonRow()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    ViewBag.border = new
    {
        width = 0,
        radius = 4,
        color = "white"
    };
    string[] xlabels = new string[9] { "China", "France", "GBR",
"Germany", "Italy", "Japan", "KOR", "Russia", "USA" };
    ViewBag.xLabels = xlabels;
}

```



```

        string[] yLabels = new string[5] { "2000", "2004", "2008",
        "2012", "2016" };
        ViewBag.yLabels = yLabels;
        ViewBag.title = new { text = "Olympic Year" };
        return View();
    }

```

JSON data - Cell binding

In JSON cell data binding, each JSON object consists a value for each cell along with a mapping value for row and column. You can bind the JSON cell data having information for each cell to the heat map using the `data` property in `dataSource`. To achieve this, you should define the `adaptorType` property as `Cell`, and enable the `isJsonData` property. Now, map the fields of data by using the `valueMapping`, `xDataMapping` and `yDataMapping` properties.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Most Visited
Destinations by International Tourist
Arrivals")).TextStyle(ViewBag.textStyle).XAxis(
xAxis => xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource("dataSource").PaletteSettings(ps
=> ps.Palette(palette =>
{
    palette.Color("#DCD57E").Add();
    palette.Color("#A6DC7E").Add();
    palette.Color("#7EDCA2").Add();
    palette.Color("#6EB5D0").Add();
})) .CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(true).Format("{value}
M")).DataSourceSettings(ds => {
ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell).XD
aMapping("rowid").YDataMapping("columnid").ValueMapping("value");
}).Render()
<script>
    // Initializing json Data
    var dataSource = [
        { 'rowid': 'France', 'columnid': '2010', 'value': '77.6' },
        { 'rowid': 'France', 'columnid': '2011', 'value': '79.4' },
        { 'rowid': 'France', 'columnid': '2012', 'value': '80.8' },
        { 'rowid': 'France', 'columnid': '2013', 'value': '86.6' },
        { 'rowid': 'France', 'columnid': '2014', 'value': '83.7' },
        { 'rowid': 'France', 'columnid': '2015', 'value': '84.5' },
        { 'rowid': 'France', 'columnid': '2016', 'value': '82.6' },
        { 'rowid': 'USA', 'columnid': '2010', 'value': '60.6' },
        { 'rowid': 'USA', 'columnid': '2011', 'value': '65.4' },
        { 'rowid': 'USA', 'columnid': '2012', 'value': '70.8' },
        { 'rowid': 'USA', 'columnid': '2013', 'value': '73.8' },
        { 'rowid': 'USA', 'columnid': '2014', 'value': '75.3' },
        { 'rowid': 'USA', 'columnid': '2015', 'value': '77.5' },
        { 'rowid': 'USA', 'columnid': '2016', 'value': '77.6' },
        { 'rowid': 'Spain', 'columnid': '2010', 'value': '64.9' },
        { 'rowid': 'Spain', 'columnid': '2011', 'value': '52.6' },
        { 'rowid': 'Spain', 'columnid': '2012', 'value': '60.8' },
        { 'rowid': 'Spain', 'columnid': '2013', 'value': '65.6' },
    ]

```

```

{ 'rowid': 'Spain', 'columnid': '2014', 'value': '52.6' },
{ 'rowid': 'Spain', 'columnid': '2015', 'value': '68.5' },
{ 'rowid': 'Spain', 'columnid': '2016', 'value': '75.6' },
{ 'rowid': 'China', 'columnid': '2010', 'value': '55.6' },
{ 'rowid': 'China', 'columnid': '2011', 'value': '52.3' },
{ 'rowid': 'China', 'columnid': '2012', 'value': '54.8' },
{ 'rowid': 'China', 'columnid': '2013', 'value': '51.1' },
{ 'rowid': 'China', 'columnid': '2014', 'value': '55.6' },
{ 'rowid': 'China', 'columnid': '2015', 'value': '56.9' },
{ 'rowid': 'China', 'columnid': '2016', 'value': '59.3' },
{ 'rowid': 'Italy', 'columnid': '2010', 'value': '43.6' },
{ 'rowid': 'Italy', 'columnid': '2011', 'value': '43.2' },
{ 'rowid': 'Italy', 'columnid': '2012', 'value': '55.8' },
{ 'rowid': 'Italy', 'columnid': '2013', 'value': '50.1' },
{ 'rowid': 'Italy', 'columnid': '2014', 'value': '48.5' },
{ 'rowid': 'Italy', 'columnid': '2015', 'value': '50.7' },
{ 'rowid': 'Italy', 'columnid': '2016', 'value': '52.4' },
{ 'rowid': 'UK', 'columnid': '2010', 'value': '28.2' },
{ 'rowid': 'UK', 'columnid': '2011', 'value': '31.6' },
{ 'rowid': 'UK', 'columnid': '2012', 'value': '29.8' },
{ 'rowid': 'UK', 'columnid': '2013', 'value': '33.1' },
{ 'rowid': 'UK', 'columnid': '2014', 'value': '32.6' },
{ 'rowid': 'UK', 'columnid': '2015', 'value': '34.4' },
{ 'rowid': 'UK', 'columnid': '2016', 'value': '35.8' },
{ 'rowid': 'Germany', 'columnid': '2010', 'value': '26.8' },
{ 'rowid': 'Germany', 'columnid': '2011', 'value': '29' },
{ 'rowid': 'Germany', 'columnid': '2012', 'value': '26.8' },
{ 'rowid': 'Germany', 'columnid': '2013', 'value': '27.6' },
{ 'rowid': 'Germany', 'columnid': '2014', 'value': '33' },
{ 'rowid': 'Germany', 'columnid': '2015', 'value': '35' },
{ 'rowid': 'Germany', 'columnid': '2016', 'value': '35.6' },
{ 'rowid': 'Mexico', 'columnid': '2010', 'value': '23.2' },
{ 'rowid': 'Mexico', 'columnid': '2011', 'value': '24.9' },
{ 'rowid': 'Mexico', 'columnid': '2012', 'value': '30.1' },
{ 'rowid': 'Mexico', 'columnid': '2013', 'value': '22.2' },
{ 'rowid': 'Mexico', 'columnid': '2014', 'value': '29.3' },
{ 'rowid': 'Mexico', 'columnid': '2015', 'value': '32.1' },
{ 'rowid': 'Mexico', 'columnid': '2016', 'value': '35' },
{ 'rowid': 'Thailand', 'columnid': '2010', 'value': '15.9' },
{ 'rowid': 'Thailand', 'columnid': '2011', 'value': '19.8' },
{ 'rowid': 'Thailand', 'columnid': '2012', 'value': '21.8' },
{ 'rowid': 'Thailand', 'columnid': '2013', 'value': '23.5' },
{ 'rowid': 'Thailand', 'columnid': '2014', 'value': '24.8' },
{ 'rowid': 'Thailand', 'columnid': '2015', 'value': '29.9' },
{ 'rowid': 'Thailand', 'columnid': '2016', 'value': '32.6' },
{ 'rowid': 'Austria', 'columnid': '2010', 'value': '22' },
{ 'rowid': 'Austria', 'columnid': '2011', 'value': '21.3' },
{ 'rowid': 'Austria', 'columnid': '2012', 'value': '24.2' },
{ 'rowid': 'Austria', 'columnid': '2013', 'value': '23.2' },
{ 'rowid': 'Austria', 'columnid': '2014', 'value': '25' },
{ 'rowid': 'Austria', 'columnid': '2015', 'value': '26.7' },
{ 'rowid': 'Austria', 'columnid': '2016', 'value': '28.1' }
];
</script>

```

```

public ActionResult JsonCell()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    ViewBag.border = new
    {
        width = 1,
        radius = 4,
        color = "white"
    };
    string[] xlabels = new string[10] { "Austria", "China", "France",
    "Germany", "Italy", "Mexico", "Spain", "Thailand", "UK", "USA" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[7] { "2010", "2011", "2012", "2013",
    "2014", "2015", "2016" };
    ViewBag.yLabels = yLabels;
    return View();
}

```

Empty points

The data points that use the **null** or **undefined** or empty string as value are considered as empty points. Empty data points are ignored and not displayed in the heat map, and these points are rendered with default palette. You can customize the empty data point color value using the [emptyPointColor](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1, 1)).
    IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years).
    LabelFormat("yyyy");
}).YAxis(yaxis =>
yaxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.Numeric)).LegendSettings(ls
=>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()

```

EMPTY-POINTS.CS

```

public ActionResult EmptyPoints()
{
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {

```

```

        [73, 39, 26, 39, 94, 0],
        [93, 58, 53, 38, 26, 68],
        [99, 28, null, 4, 66, 90],
        [14, 26, 97, 69, 69, 3],
        [7, 46, 47, 47, 88, 6],
        [41, 55, 73, 23, "", 79],
        [56, 69, 21, 86, 3, 33],
        [45, 7, undefined, 81, 95, 79],
        [60, 77, 74, 68, 88, 51],
        [25, 25, 10, 12, 78, 14],
        [25, 56, 55, 58, 12, 82],
        [74, 33, 88, 23, 86, 59]
    ];
    return data;
}

```

Binding nested JSON data

In complex data binding, you can bind the nested JSON data to the data points in the heat map. The nested data can be mapped using the `xDataMapping`, `yDataMapping`, `valueMapping` and `bubbleDataMapping` properties as string value concatenated by a dot.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Most Visited
Destinations by International Tourist
Arrivals")).TextStyle(ViewBag.textStyle)).XAxis(
xAxis => xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource("dataSource").PaletteSettings(ps
=> ps.Palette(palette =>
{
    palette.Color("#DCD57E").Add();
    palette.Color("#A6DC7E").Add();
    palette.Color("#7EDCA2").Add();
    palette.Color("#6EB5D0").Add();
})).CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(true).Format("{value}
M")).DataSourceSettings(ds => {
ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell).XDat
aMapping("Labels.Xlabel").YDataMapping("Labels.Ylabel").ValueMapping("data.
value"); })Render()
<script>
    var load = function (args) {
        var selectedTheme = location.hash.split('/')[1];
        selectedTheme = selectedTheme ? selectedTheme : 'Material';
        args.heatmap.theme = (selectedTheme.charAt(0).toUpperCase() +
selectedTheme.slice(1));
    };
    var dataSource = [
{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2010' }, 'data':{ 'value': '77.6'
}},
{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2011' }, 'data':{ 'value': '79.4' }},
{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2012' }, 'data':{ 'value': '80.8' }},
{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2013' }, 'data':{ 'value': '86.6' }},
{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2014' }, 'data':{ 'value': '83.7' }},
{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2015' }, 'data':{ 'value': '84.5' }},

```

```

{ 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2016'}, 'data':{ 'value': '82.6'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2010'}, 'data':{ 'value': '60.6'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2011'}, 'data':{ 'value': '65.4'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2012'}, 'data':{ 'value': '70.8'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2013'}, 'data':{ 'value': '73.8'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2014'}, 'data':{ 'value': '75.3'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2015'}, 'data':{ 'value': '77.5'}},
{ 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2016'}, 'data':{ 'value': '77.6'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2010'}, 'data':{ 'value': '64.9'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2011'}, 'data':{ 'value': '52.6'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2012'}, 'data':{ 'value': '60.8'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2013'}, 'data':{ 'value': '65.6'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2014'}, 'data':{ 'value': '52.6'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2015'}, 'data':{ 'value': '68.5'}},
{ 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2016'}, 'data':{ 'value': '75.6'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2010'}, 'data':{ 'value': '55.6'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2011'}, 'data':{ 'value': '52.3'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2012'}, 'data':{ 'value': '54.8'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2013'}, 'data':{ 'value': '51.1'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2014'}, 'data':{ 'value': '55.6'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2015'}, 'data':{ 'value': '56.9'}},
{ 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2016'}, 'data':{ 'value': '59.3'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2010'}, 'data':{ 'value': '43.6'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2011'}, 'data':{ 'value': '43.2'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2012'}, 'data':{ 'value': '55.8'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2013'}, 'data':{ 'value': '50.1'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2014'}, 'data':{ 'value': '48.5'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2015'}, 'data':{ 'value': '50.7'}},
{ 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2016'}, 'data':{ 'value': '52.4'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2010'}, 'data':{ 'value': '28.2'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2011'}, 'data':{ 'value': '31.6'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2012'}, 'data':{ 'value': '29.8'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2013'}, 'data':{ 'value': '33.1'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2014'}, 'data':{ 'value': '32.6'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2015'}, 'data':{ 'value': '34.4'}},
{ 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2016'}, 'data':{ 'value': '35.8'}},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2010'}, 'data':{ 'value': '26.8'}},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2011'}, 'data':{ 'value': '29' }},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2012'}, 'data':{ 'value': '26.8'}},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2013'}, 'data':{ 'value': '27.6'}},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2014'}, 'data':{ 'value': '33' }},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2015'}, 'data':{ 'value': '35' }},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2016'}, 'data':{ 'value': '35.6'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2010'}, 'data':{ 'value': '23.2'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2011'}, 'data':{ 'value': '24.9'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2012'}, 'data':{ 'value': '30.1'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2013'}, 'data':{ 'value': '22.2'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2014'}, 'data':{ 'value': '29.3'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2015'}, 'data':{ 'value': '32.1'}},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2016'}, 'data':{ 'value': '35' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2010'}, 'data':{ 'value':
'15.9'}},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2011'}, 'data':{ 'value':
'19.8'}},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2012'}, 'data':{ 'value':
'21.8'}},

```

```

{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2013'}, 'data':{ 'value':
'23.5' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2014'}, 'data':{ 'value':
'24.8' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2015'}, 'data':{ 'value':
'29.9' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2016'}, 'data':{ 'value':
'32.6' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2010'}, 'data':{ 'value': '22' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2011'}, 'data':{ 'value': '21.3' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2012'}, 'data':{ 'value': '24.2' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2013'}, 'data':{ 'value': '23.2' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2014'}, 'data':{ 'value': '25' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2015'}, 'data':{ 'value': '26.7'
}},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2016'}, 'data':{ 'value': '28.1' }}
];
</script>

```

NESTED-JSON-CELL.CS

```

public ActionResult JsonCell()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    ViewBag.border = new
    {
        width = 1,
        radius = 4,
        color = "white"
    };
    string[] xlabels = new string[10] { "Austria", "China", "France",
    "Germany", "Italy", "Mexico", "Spain", "Thailand", "UK", "USA" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[7] { "2010", "2011", "2012", "2013",
    "2014", "2015", "2016" };
    ViewBag.yLabels = yLabels;
    return View();
}

```

See Also

- [To bind data for bubble heat map with size and color attributes](#)

Bubble HeatMap in ASP.NET MVC HeatMap Chart Component

The cell customizations and color mapping for rect tile type is defined in [appearance](#) and [palette](#) sections in detail.

Bubble types

In bubble HeatMap, you can display the data points with bubble size, bubble colors, and sector attributes of the bubble.

Bubble size

In this bubble HeatMap type, the size factor of the bubble is used to denote the data variations. The radius of the bubble varies according to data values.

By default, the bubble with small size denotes the data value with small magnitude and the larger bubble size denotes the data value with larger magnitude. This behavior can be inverted by using the [IsInversedbubblesize](#) property.

To render a bubble HeatMap with size series, set the [BubbleType](#) property to **Size**.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
cs.ShowLabel(false).Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.C
ellType.Bubble).BubbleType(Syncfusion.EJ2.HeatMap.BubbleType.Size)).DataSou
rce(ViewBag.dataSource).Render()
```

SIZE-BUBBLE.CS

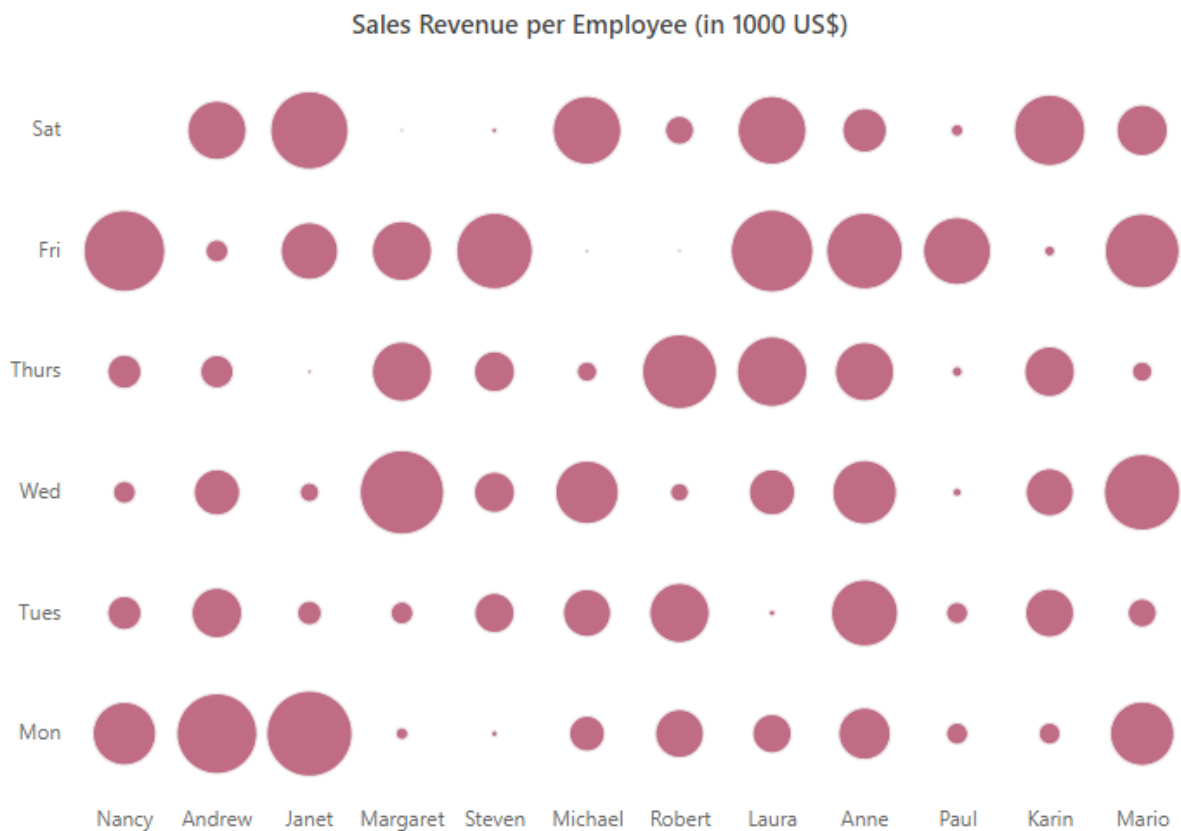
```
public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{

```

```

int[,] data = new int[,]
{
    {73, 39, 26, 39, 94, 0},
    {93, 58, 53, 38, 26, 68},
    {99, 28, 22, 4, 66, 90},
    {14, 26, 97, 69, 69, 3},
    {7, 46, 47, 47, 88, 6},
    {41, 55, 73, 23, 3, 79},
    {56, 69, 21, 86, 3, 33},
    {45, 7, 53, 81, 95, 79},
    {60, 77, 74, 68, 88, 51},
    {25, 25, 10, 12, 78, 14},
    {25, 56, 55, 58, 12, 82},
    {74, 33, 88, 23, 86, 59}
};
return data;
}

```



Bubble color

In HeatMap, defined with this tile type, the data points will be represented with same sized bubbles and the data value variations are represented with the bubble colors.

To represent the data points with variations in bubble colors, set the [BubbleType](#) property to **Color**.

CSHTML

```
@using Syncfusion.EJ2;
```

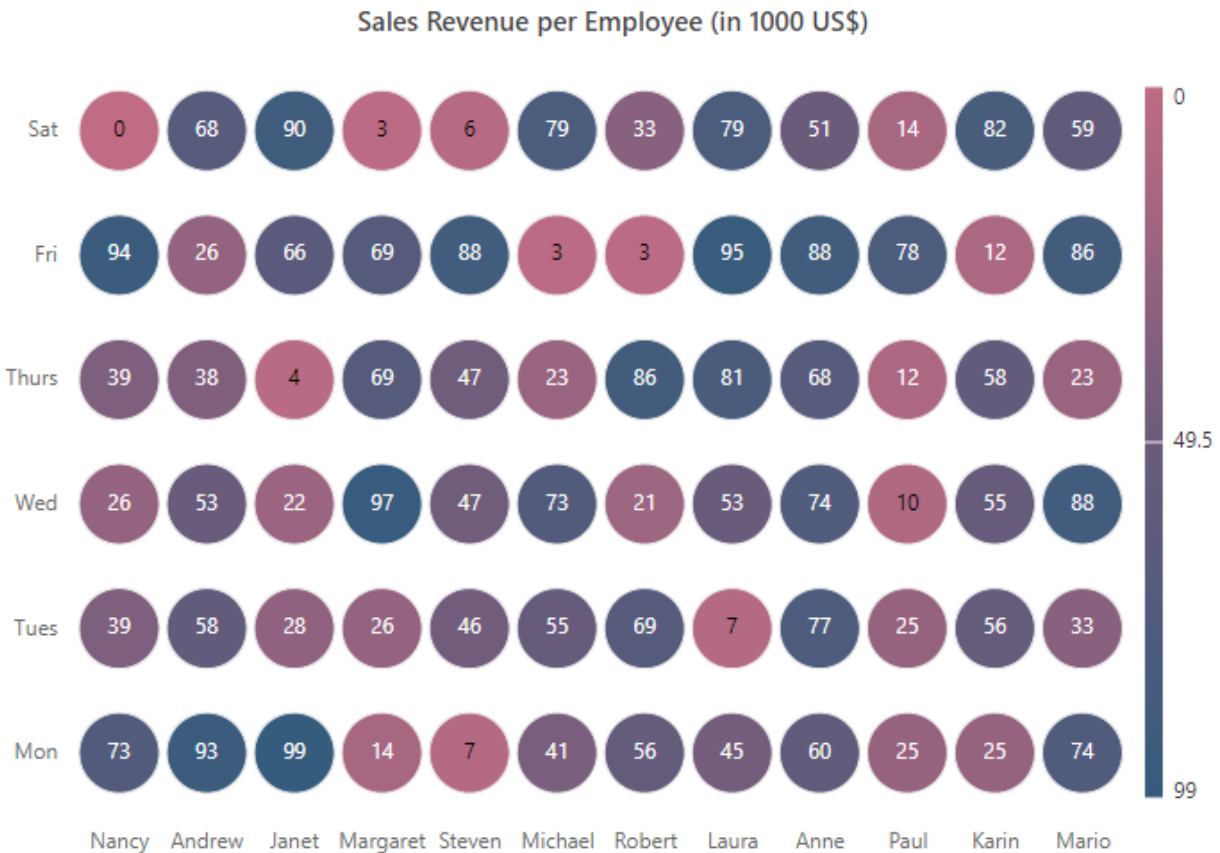


```
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>

cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).B
ubbleType(Syncfusion.EJ2.HeatMap.BubbleType.Color)).DataSource(ViewBag.dataS
ource).Render()
```

COLOR-BUBBLE.CS

```
public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```



Bubble sector

In this bubble HeatMap type, the sector of the bubble decides the magnitude of data point. If the sector is large, then the data point value will be high.

To render the data points with bubble sector, set the [BubbleType](#) property to **Sector**.

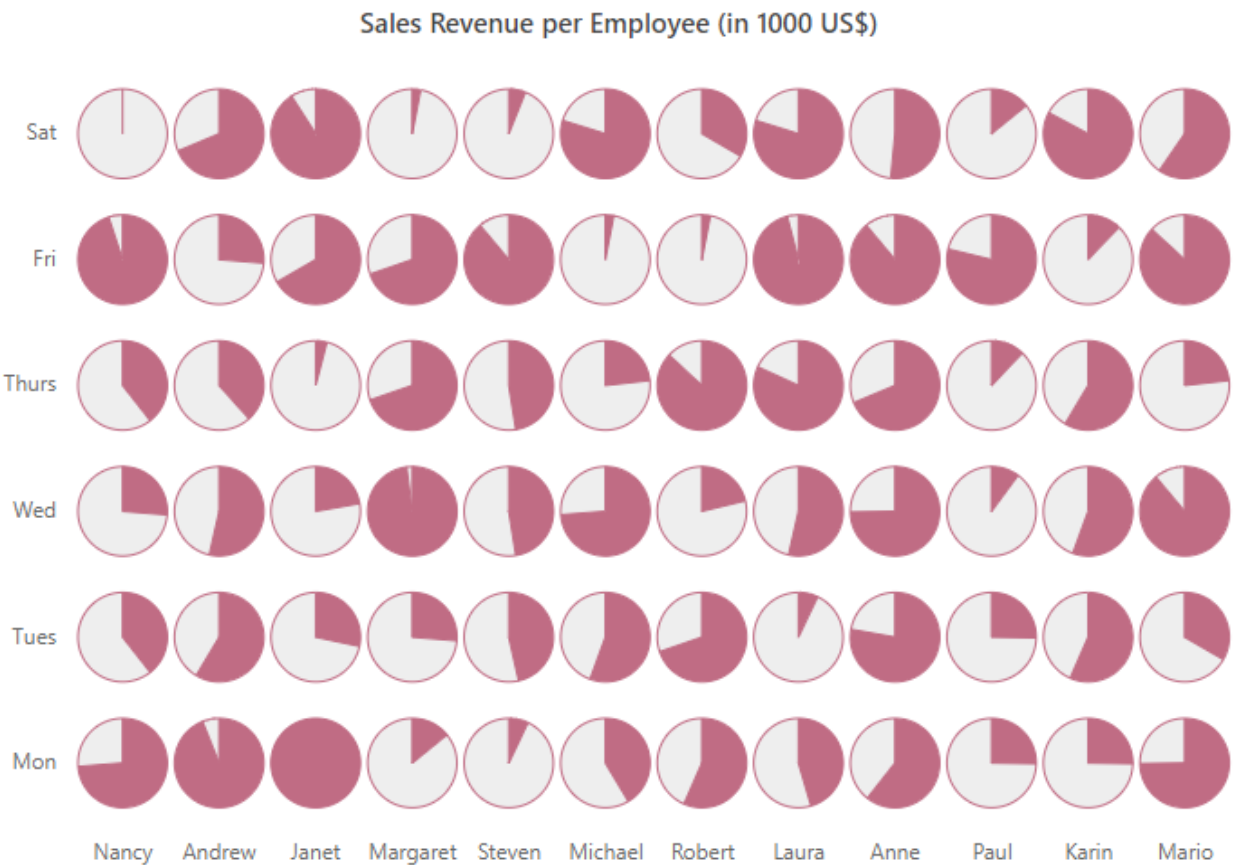
CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).B
ubbleType(Syncfusion.EJ2.HeatMap.BubbleType.Sector)).DataSource(ViewBag.data
Source).Render()
```

SECTOR-BUBBLE.CS

```
public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```



Combining size and color bubble types

The following examples demonstrate different data binding with the **SizeAndColor** bubble type set in the HeatMap.

<!-- markdownlint-disable MD036 -->

Array binding

When an array of numbers is specified as the data source, the bubble HeatMap can be rendered with different sizes and colors depending on the bound data.

<!-- markdownlint-disable MD036 -->

Table

The following example illustrates how to render a bubble HeatMap with different sizes and colors using array table binding.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2012 - 2017").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
    ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette
=> {
```

```

        palette.Color("#C06C84").Add();
        palette.Color("#6C5B7B").Add();
        palette.Color("#355C7D").Add();

    }).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>

cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).B
ubbleType(Syncfusion.EJ2.HeatMap.BubbleType.SizeAndColor)).DataSource("dataS
ource").TooltipRender("tooltipRender").Render()
}
<script>
    var tooltipRender = function (args) {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
            + 'Accidents ' + ' : ' + args.value[0].bubbleData + '<br/>' +
'Fatalities ' + ' : '
            + args.value[1].bubbleData];
    };
    var dataSource = [
        [[4, 39], [3, 8], [1, 3], [1, 10], [4, 4], [2, 15]],
        [[4, 28], [5, 92], [5, 73], [3, 1], [3, 4], [4, 126]],
        [[4, 45], [5, 152], [0, 44], [4, 54], [5, 243], [2, 45]]
    ];
</script>

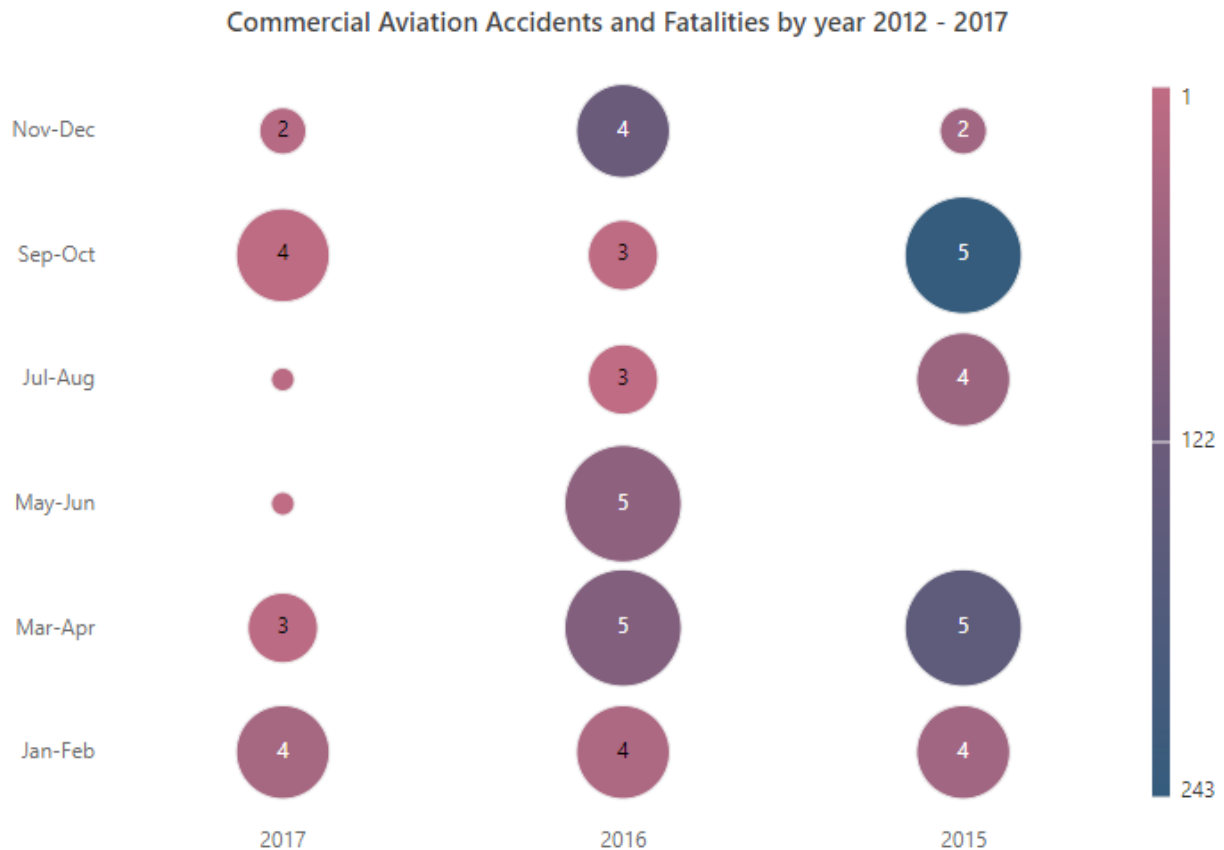
```

SIZE-COLOR-TABLE.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    ViewBag.border = new
    {
        color = "red",
        opacity = 1
    };
    string[] xlabels = new string[3] { "2017", "2016", "2015" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-
Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```



<!-- markdownlint-disable MD036 -->

Cell

The following example illustrates how to render a bubble HeatMap with different sizes and colors using array cell binding.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2012 - 2017").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
                ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette
=> {
            palette.Color("#C06C84").Add();
            palette.Color("#6C5B7B").Add();
            palette.Color("#355C7D").Add();
        })
    ).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
    cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).BubbleType(Syncfusion.EJ2.HeatMap.BubbleType.SizeAndColor).DataSource("dataSource").TooltipRender("tooltipRender").DataSourceSettings(ds => {
```

```

ds.IsJsonData(false).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell);
}).Render()
<script>
    var tooltipRender = function (args) {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
            + 'Accidents ' + ' : ' + args.value[0].bubbleData + '<br/>' +
'Fatalities ' + ' : '
            + args.value[1].bubbleData];
    };
    var dataSource = [
        [0,0,[4,39]], [0,1,[3,8]], [0,2,[1,3]], [0,3,[1,10]], [0,4,[4,4]],
[0,5,[2,15]],
        [1,0,[4,28]], [1,1,[5,92]], [1,2,[5,73]], [1,3,[3,1]], [1,4,[3,4]],
[1,5,[4,126]],
        [2,0,[4,45]], [2,1,[5,152]], [2,2,[0,44]], [2,3,[4,54]], [2,4,[5,243]],
[2,5,[2,45]]
    ];
</script>

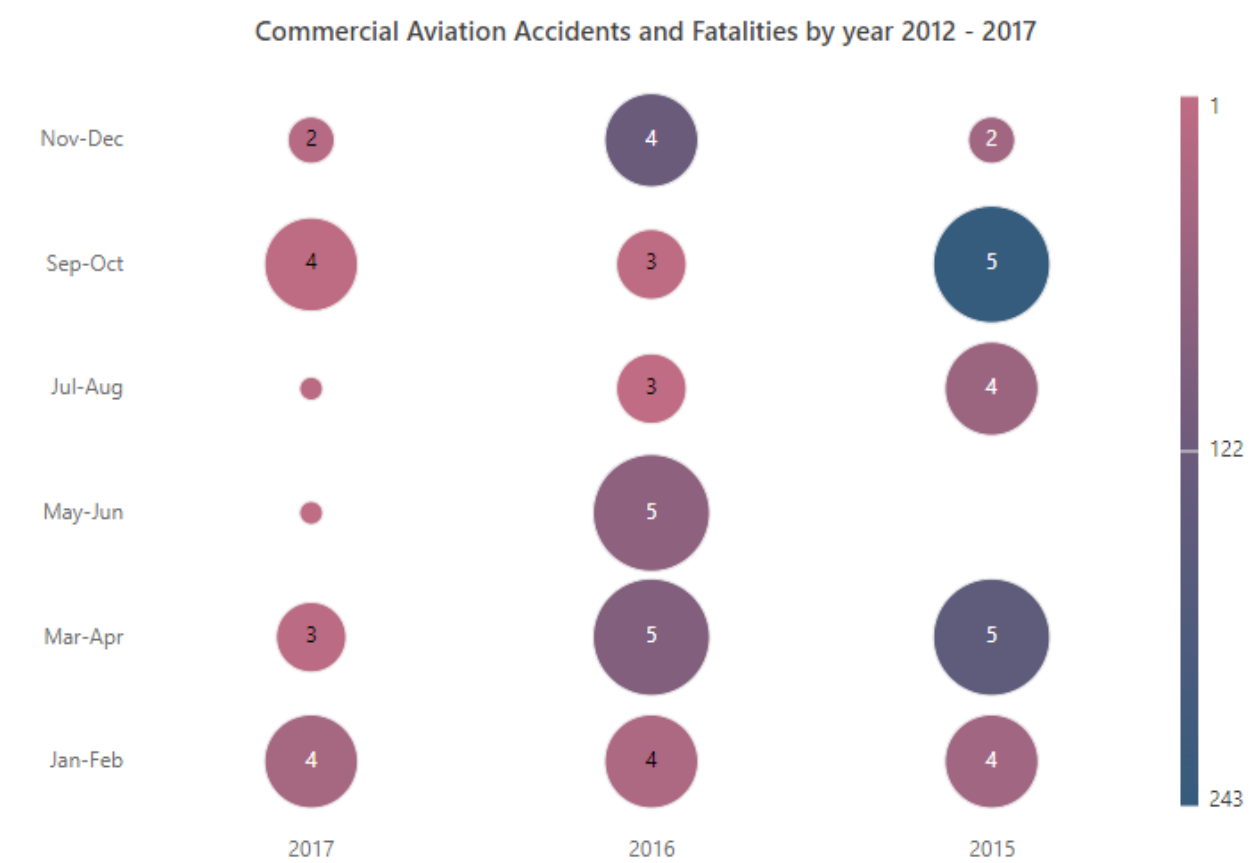
```

SIZE-COLOR-CELL.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    ViewBag.border = new
    {
        color = "red",
        opacity = 1
    };
    string[] xlabels = new string[3] { "2017", "2016", "2015" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-
Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```



<!-- markdownlint-disable MD036 -->

JSON binding

When a list of JSON objects are specified as data source, the bubble HeatMap can be rendered with different sizes and colors depending on the bound data.

<!-- markdownlint-disable MD036 -->

Table

The following example illustrates how to render a bubble HeatMap with different sizes and colors using JSON table binding.

CSTHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2012 - 2017").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
    ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette
=> {
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
```



```

}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).B
ubbleType(Syncfusion.EJ2.HeatMap.BubbleType.SizeAndColor)).DataSource("dataS
ource").TooltipRender("tooltipRender").DataSourceSettings(ds => {
ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Table).XD
ataMapping("Year"); }).Render()
<script>
    var tooltipRender = function (args) {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
            + 'Accidents ' + ' : ' + args.value[0].bubbleData + '<br/>' +
'Fatalities ' + ' : '
            + args.value[1].bubbleData];
    };
    var dataSource = [
        {'Year': '2017', 'Jan-Feb': [4,39], 'Mar-Apr': [3,8], 'May-Jun': [1,3],
'Jul-Aug': [1,10], 'Sep-Oct': [4,4], 'Nov-Dec': [2,15]},
        {'Year': '2016', 'Jan-Feb': [4,28], 'Mar-Apr': [5,92], 'May-Jun':
[5,73], 'Jul-Aug': [3,1], 'Sep-Oct': [3,4], 'Nov-Dec': [4,126]},
        {'Year': '2015', 'Jan-Feb': [4,45], 'Mar-Apr': [5,152], 'May-Jun':
[0,44], 'Jul-Aug': [4,54], 'Sep-Oct': [5,243], 'Nov-Dec': [2,45]},
    ];
</script>

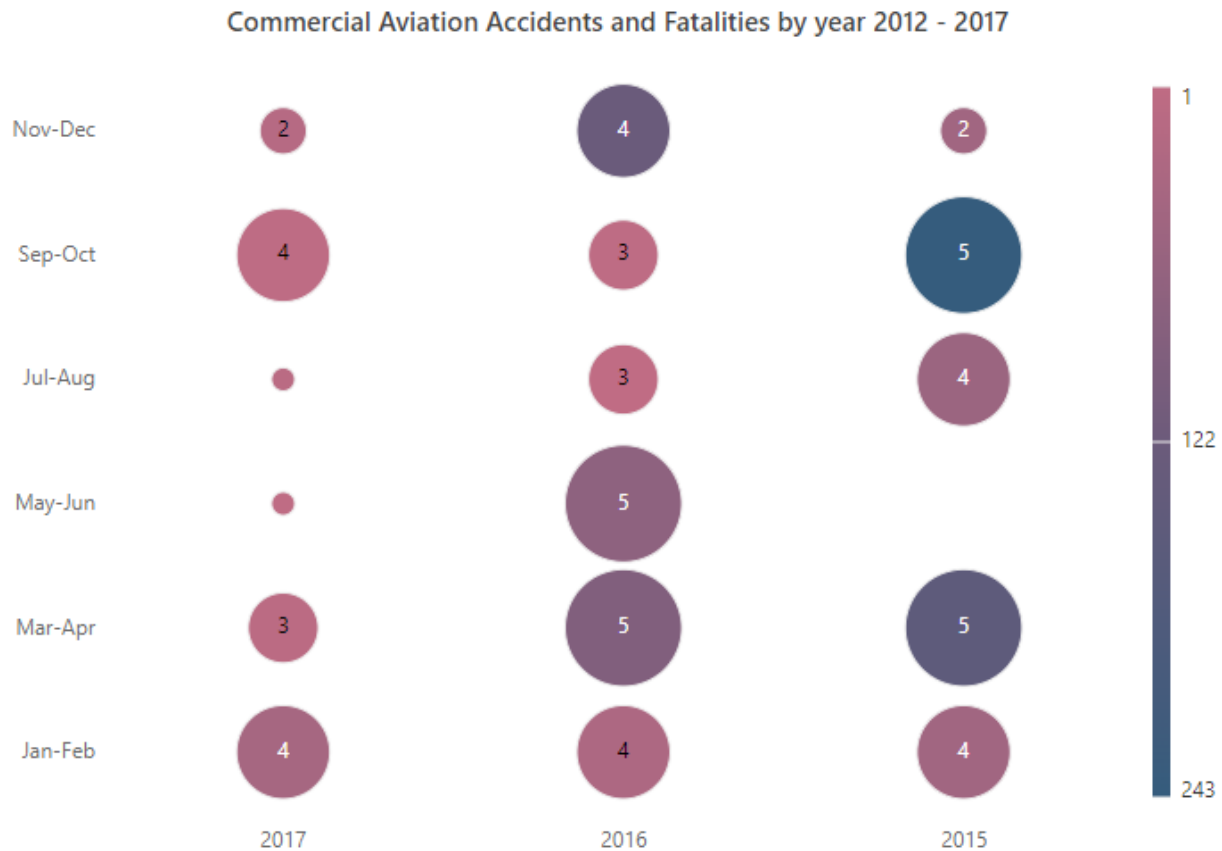
```

SIZE-COLOR-JSON-TABLE.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    ViewBag.border = new
    {
        color = "red",
        opacity = 1
    };
    string[] xlabels = new string[3] { "2017", "2016", "2015" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-
Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```



<!-- markdownlint-disable MD036 -->

Cell

The following example illustrates how to render a bubble HeatMap with different sizes and colors using JSON cell binding.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2012 - 2017").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
    ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette
=> {
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).BubbleType(Syncfusion.EJ2.HeatMap.BubbleType.SizeAndColor).DataSource("dataSource").TooltipRender("tooltipRender").DataSourceSettings(ds => {
ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell)).XDa
```

```

taMapping("Year").YDataMapping("Months").BubbleDataMapping(bm =>
bm.Size("Accidents").Color("Fatalities")); }).Render()
</script>
var tooltipRender = function (args) {
    args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months ' + '
: ' + args.yLabel + '<br/>'
        + 'Accidents ' + ' : ' + args.value[0].bubbleData + '<br/>' +
'Fatalities ' + ' : '
        + args.value[1].bubbleData];
};
var dataSource = [
    { Year: '2017', Months: 'Jan-Feb', Accidents: 4, Fatalities: 39 },
    { Year: '2017', Months: 'Mar-Apr', Accidents: 3, Fatalities: 8 },
    { Year: '2017', Months: 'May-Jun', Accidents: 1, Fatalities: 3 },
    { Year: '2017', Months: 'Jul-Aug', Accidents: 1, Fatalities: 10 },
    { Year: '2017', Months: 'Sep-Oct', Accidents: 4, Fatalities: 4 },
    { Year: '2017', Months: 'Nov-Dec', Accidents: 2, Fatalities: 15 },
    { Year: '2016', Months: 'Jan-Feb', Accidents: 4, Fatalities: 28 },
    { Year: '2016', Months: 'Mar-Apr', Accidents: 5, Fatalities: 92 },
    { Year: '2016', Months: 'May-Jun', Accidents: 5, Fatalities: 73 },
    { Year: '2016', Months: 'Jul-Aug', Accidents: 3, Fatalities: 1 },
    { Year: '2016', Months: 'Sep-Oct', Accidents: 3, Fatalities: 4 },
    { Year: '2016', Months: 'Nov-Dec', Accidents: 4, Fatalities: 126 },
    { Year: '2015', Months: 'Jan-Feb', Accidents: 4, Fatalities: 45 },
    { Year: '2015', Months: 'Mar-Apr', Accidents: 5, Fatalities: 152 },
    { Year: '2015', Months: 'May-Jun', Accidents: 0, Fatalities: 0 },
    { Year: '2015', Months: 'Jul-Aug', Accidents: 4, Fatalities: 54 },
    { Year: '2015', Months: 'Sep-Oct', Accidents: 5, Fatalities: 243 },
    { Year: '2015', Months: 'Nov-Dec', Accidents: 2, Fatalities: 45 },
];
</script>

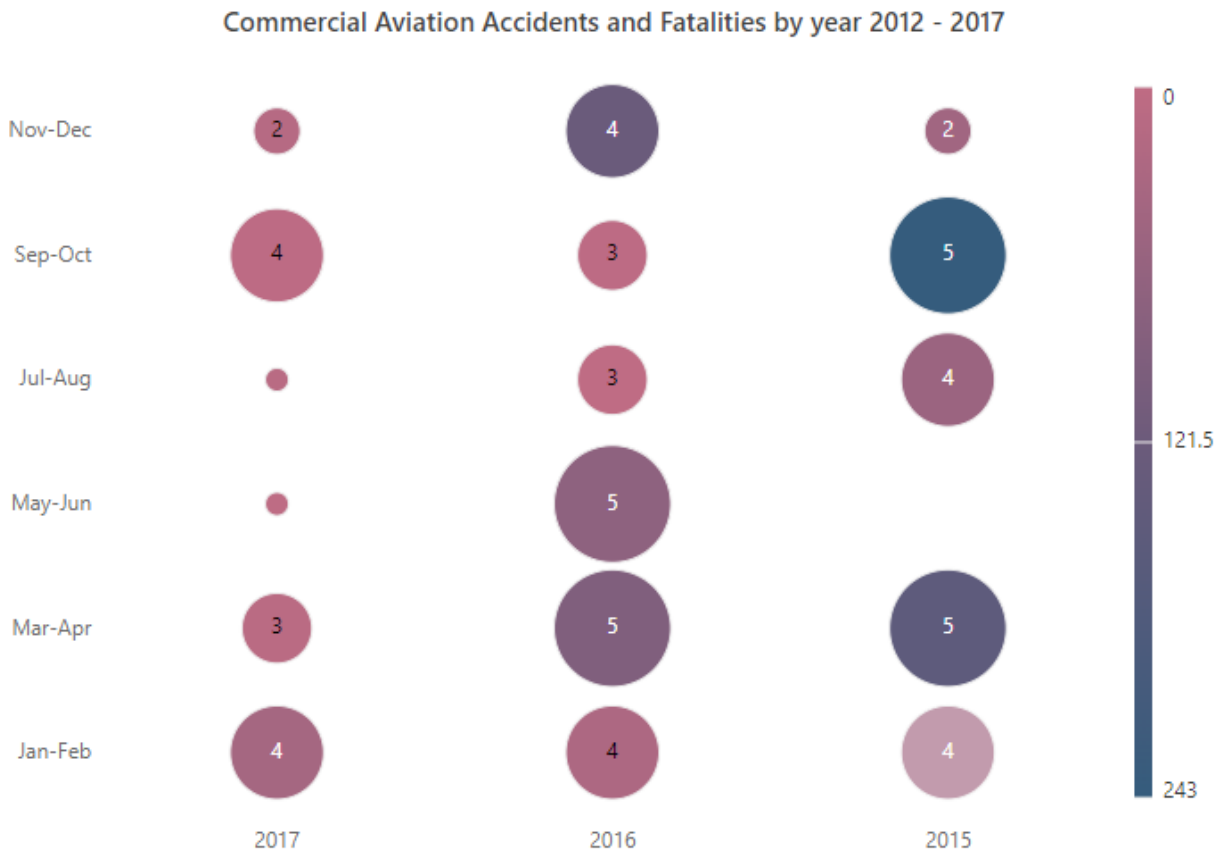
```

SIZE-COLOR-JSON-CELL.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    ViewBag.border = new
    {
        color = "red",
        opacity = 1
    };
    string[] xlabels = new string[3] { "2017", "2016", "2015" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-
Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```



<!-- markdownlint-disable MD036 -->

Binding size and color values from datasource

The size and color of the bubbles in the **SizeAndColor** bubble HeatMap type can be customized by binding the datasource field name that holds the size and color values to the [Size](#) and [Color](#) properties in the [BubbleDataMapping](#).

The **BubbleDataMapping** supports only for the JSON data with cell adaptor type.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2012 - 2017").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
                ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette
=> {
                palette.Color("#C06C84").Add();
                palette.Color("#6C5B7B").Add();
                palette.Color("#355C7D").Add();
            })
        ).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
        cs.Border(ViewBag.border).TileType(Syncfusion.EJ2.HeatMap.CellType.Bubble).B
```

```

bubbleType(Syncfusion.EJ2.HeatMap.BubbleType.SizeAndColor)).DataSource("dataSource").TooltipRender("tooltipRender").DataSourceSettings(ds => {
    ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell).XDataMapping("Year").YDataMapping("Months").BubbleDataMapping(bm =>
        bm.Size("Accidents").Color("Fatalities")); }).Render()

<script>
var tooltipRender = function (args) {
    args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months ' + ' : ' + args.yLabel + '<br/>'
        + 'Accidents ' + ' : ' + args.value[0].bubbleData + '<br/>' + 'Fatalities ' + ' : ' + args.value[1].bubbleData];
};
var dataSource = [
    { Year: '2017', Months: 'Jan-Feb', Accidents: 4, Fatalities: 39 },
    { Year: '2017', Months: 'Mar-Apr', Accidents: 3, Fatalities: 8 },
    { Year: '2017', Months: 'May-Jun', Accidents: 1, Fatalities: 3 },
    { Year: '2017', Months: 'Jul-Aug', Accidents: 1, Fatalities: 10 },
    { Year: '2017', Months: 'Sep-Oct', Accidents: 4, Fatalities: 4 },
    { Year: '2017', Months: 'Nov-Dec', Accidents: 2, Fatalities: 15 },
    { Year: '2016', Months: 'Jan-Feb', Accidents: 4, Fatalities: 28 },
    { Year: '2016', Months: 'Mar-Apr', Accidents: 5, Fatalities: 92 },
    { Year: '2016', Months: 'May-Jun', Accidents: 5, Fatalities: 73 },
    { Year: '2016', Months: 'Jul-Aug', Accidents: 3, Fatalities: 1 },
    { Year: '2016', Months: 'Sep-Oct', Accidents: 3, Fatalities: 4 },
    { Year: '2016', Months: 'Nov-Dec', Accidents: 4, Fatalities: 126 },
    { Year: '2015', Months: 'Jan-Feb', Accidents: 4, Fatalities: 45 },
    { Year: '2015', Months: 'Mar-Apr', Accidents: 5, Fatalities: 152 },
    { Year: '2015', Months: 'May-Jun', Accidents: 0, Fatalities: 0 },
    { Year: '2015', Months: 'Jul-Aug', Accidents: 4, Fatalities: 54 },
    { Year: '2015', Months: 'Sep-Oct', Accidents: 5, Fatalities: 243 },
    { Year: '2015', Months: 'Nov-Dec', Accidents: 2, Fatalities: 45 },
];
</script>

```

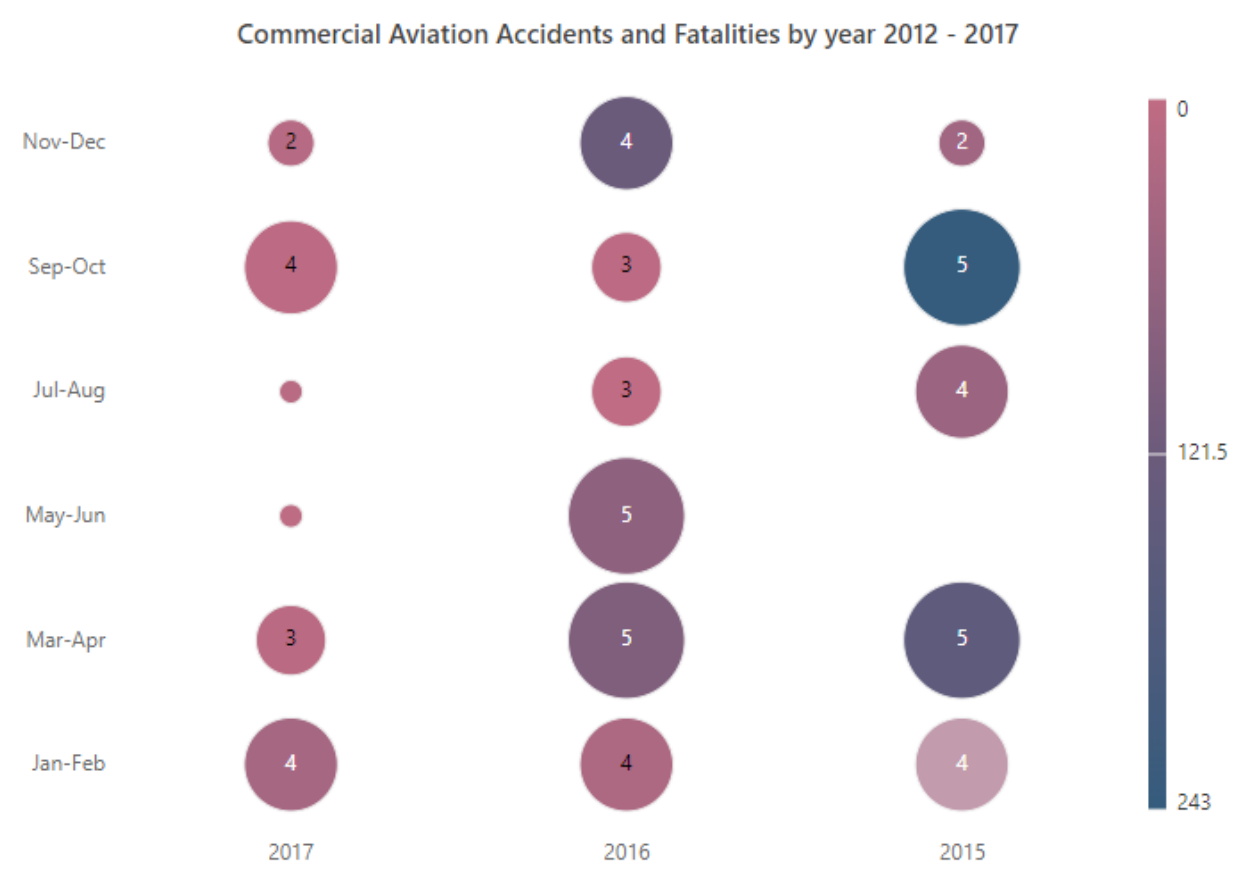
SIZE-COLOR-JSON-CELL.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    ViewBag.border = new
    {
        color = "red",
        opacity = 1
    };
    string[] xlabels = new string[3] { "2017", "2016", "2015" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
}

```

```
        return View();
    }
```



Rendering mode in ASP.NET MVC HeatMap Chart Component

Heat map can be displayed using **Canvas** or **Scalable Vector Graphics (SVG)** rendering logic to improve the initial load performance and scalability. Heat map can also be automatically switched between **Canvas** and **SVG** modes based on dataset size. You can enable this mode by setting the [renderingMode](#) property to **Auto**.

Note: If the **Auto** mode is enabled in the heat map and there are more than 10,000 data points, then the heat map will be rendered in a **Canvas** mode; Otherwise, the heat map will be rendered in a **SVG** mode.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000 US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(true)).RenderingMode(Syncfusion.EJ2.HeatMap.DrawType.SVG).ShowTooltip(true).DataSource(ViewBag.dataSource).Render()
```

RENDERING-MODE.CS

```

public ActionResult RenderingMode()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}

```

Axis in ASP.NET MVC HeatMap Chart Component

HeatMap consists of two axes namely, X-axis and Y-axis that displays the row headers and column headers to plot the data points respectively. You can define the type, format, and other customizing options for both axes in the HeatMap.

Types**Category axis**

Category axis type is used to represent the string values in axis labels.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>

```

```

xAxis.Labels(ViewBag.xLabels).ValueType(Syncfusion.EJ2.HeatMap.ValueType.Category)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels).ValueType(Syncfusion.EJ2.HeatMap.ValueType.Category)).DataSource(ViewBag.dataSource).Render()

```

CATEGORY.CS

```

public ActionResult Category()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}

```

Numeric axis

Numeric axis type is used to represent the numeric values in axis labels.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000 US$)").XAxis(xAxis =>
xAxis.Minimum(1).Maximum(11).ValueType(Syncfusion.EJ2.HeatMap.ValueType.Numeric)).YAxis(yaxis =>

```



```
yaxis.Minimum(1).Maximum(5).ValueType(Syncfusion.EJ2.HeatMap.ValueType.Category).DataSource(ViewBag.dataSource).Render()
```

NUMERIC.CS

```
public ActionResult Numeric()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}
```

Date-time axis

Date-time axis type is used to represent the date-time values in axis labels with a specific format. In date-time axis, you can define the start and end date/time using the [Minimum](#) and [Maximum](#) properties.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Monthly Flight Traffic at JFK
Airport").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1, 1))
    .Maximum(new System.DateTime(2017, 1,
1)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years)
    .LabelFormat("yyyy").LabelRotation(45)
    .LabelIntersectAction(Syncfusion.EJ2.HeatMap.LabelIntersectAction.None);
})
```

```

}).YAxis(yaxis => yaxis.Labels( ViewBag.yLabels)).CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(false).Format("{value}
flights")).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()

```

DATETIME.CS

```

public ActionResult DateTime()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] yLabels = new string[12] { "Jan", "Feb", "Mar", "Apr",
    "May", "June", "July", "Aug",
    "Sep", "Oct", "Nov", "Dec"};
    ViewBag.yLabels = yLabels;
    ViewBag.border = new { width = 0};
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964,
36543, 30529, 33298, 36985},
            {34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467,
35390, 34196, 35302, 35703},
            {34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255,
32776, 32645, 31539, 32981},
            {32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837,
32910, 33437, 30659, 31965},
            {31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497,
34285, 34094, 32256, 33699},
            {31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943,
34128, 30624, 32398, 33522},
            {32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301,
33627, 34115, 31072, 33939},
            {32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978,
35777, 35277, 31281, 35411},
            {32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335,
37177, 37443, 32457, 37304},
            {34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868,
38252, 36505, 29576, 36450},
            {35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002,
37021, 36583, 32408, 37108}
        }
    };
    return data;
}

```

Inversed axis

HeatMap supports inverting the axis origin for both axes, where the axis labels are placed in an inversed manner. You can enable axis inverting by enabling the [IsInversed](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Monthly Flight Traffic at JFK
Airport").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1, 1))
    .Maximum(new System.DateTime(2017, 1,
1)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years)
    .LabelFormat("yyyy").LabelRotation(45)

.LabelIntersectAction(Syncfusion.EJ2.HeatMap.LabelIntersectAction.None).IsIn
versed(true);
}).YAxis(yaxis => yaxis.Labels( ViewBag.yLabels)).CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(false).Format("{value}
flights")).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()
```

INVERSED.CS

```
public ActionResult Inversed()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] yLabels = new string[12] { "Jan", "Feb", "Mar", "Apr",
"May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"};
    ViewBag.yLabels = yLabels;
    ViewBag.border = new { width = 0};
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964,
36543, 30529, 33298, 36985},
        {34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467,
35390, 34196, 35302, 35703},
        {34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255,
32776, 32645, 31539, 32981},
        {32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837,
32910, 33437, 30659, 31965},
    }
```

```

        {31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497,
34285, 34094, 32256, 33699},
        {31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943,
34128, 30624, 32398, 33522},
        {32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301,
33627, 34115, 31072, 33939},
        {32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978,
35777, 35277, 31281, 35411},
        {32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335,
37177, 37443, 32457, 37304},
        {34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868,
38252, 36505, 29576, 36450},
        {35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002,
37021, 36583, 32408, 37108}
    };
    return data;
}

```

Opposed axis

In HeatMap, you can place the axis label in an opposite position of its default axis label position by using the [OpposedPosition](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Monthly Flight Traffic at JFK
Airport").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1, 1))
    .Maximum(new System.DateTime(2017, 1,
1)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years)
    .LabelFormat("yyyy").LabelRotation(45)

    .LabelIntersectAction(Syncfusion.EJ2.HeatMap.LabelIntersectAction.None).Oppo
sedPosition(true);
}).YAxis(yaxis => yaxis.Labels( ViewBag.yLabels)).CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(false).Format("{value}
flights")).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()

```

OPPOSED.CS

```

public ActionResult Opposed()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] yLabels = new string[12] { "Jan", "Feb", "Mar", "Apr",
"May", "June", "July", "Aug",

```

```

        "Sep", "Oct", "Nov", "Dec"};
        ViewBag.yLabels = yLabels;
        ViewBag.border = new { width = 0};
        ViewBag.dataSource = GetDataSource();
        return View();
    }

    private int[,] GetDataSource()
    {
        int[,] data = new int[,]{
            {36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964,
36543, 30529, 33298, 36985},
            {34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467,
35390, 34196, 35302, 35703},
            {34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255,
32776, 32645, 31539, 32981},
            {32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837,
32910, 33437, 30659, 31965},
            {31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497,
34285, 34094, 32256, 33699},
            {31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943,
34128, 30624, 32398, 33522},
            {32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301,
33627, 34115, 31072, 33939},
            {32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978,
35777, 35277, 31281, 35411},
            {32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335,
37177, 37443, 32457, 37304},
            {34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868,
38252, 36505, 29576, 36450},
            {35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002,
37021, 36583, 32408, 37108}
        };
        return data;
    }
}

```

Axis labels customization

Customizing the text style

The text style of the axis labels can be customized using the following options available in the [TextStyle](#) property.

- [Color](#) - It is used to change the text color of the axis labels.
- [FontFamily](#) - It is used to change the font family of the axis labels.
- [FontStyle](#) - It is used to change the font style of the axis labels.
- [FontWeight](#) - It is used to change the font weight of the axis labels.
- [Size](#) - It is used to change the font size of the axis labels.
- [TextAlignment](#) - It is used to position and align the axis labels. This property allows you to specify values such as **Near**, **Center**, and **Far**.
- [TextOverflow](#) - When the axis label exceeds the intended space, this property is used to trim or wrap it. This property takes values such as **None**, **Trim**, and **Wrap**.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Load("heatmapLoad").TitleSettings(ts =>
ts.Text("Product wise Monthly sales revenue for a e-commerce website").
    TextStyle(ViewBag.textStyle)).XAxis(
    xAxis =>
xAxis.Labels(ViewBag.xLabels).OpposedPosition(true).TextStyle(ViewBag.axisTextStyle)).YAxis(yaxis =>

yaxis.Labels(ViewBag.yLabels).TextStyle(ViewBag.axisTextStyle).MaxLabelLength(70)).PaletteSettings(ps => ps.Palette(palette => {
    palette.Color("#F0C27B").Add();
    palette.Color("#4B1248").Add();
}))).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapLoad(args) {
        args.heatmap.xAxis.textStyle.textAlignment = "Center";
        args.heatmap.yAxis.textStyle.textAlignment = "Center";
        args.heatmap.xAxis.textStyle.textOverflow = "Wrap";
        args.heatmap.yAxis.textStyle.textOverflow = "Wrap";
    }
</script>

```

TEXTSTYLE.CS

```

public ActionResult TextStyle()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    ViewBag.axisTextStyle = new
    {
        color="Red",
        size= "15px",
        fontWeight= "650",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[] { "Month of Feburary 2023", "Month of March 2023", "Month of April 2023", "Month of May 2023", "Month of June 2023", "Month of July 2023", "Month of August 2023", "Month of September 2023", "Month of October 2023", "Month of November 2023", "Month of December 2023" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[] { "Ace Apparels", "Alpha Apparels", "RL Garments", "RRD Garments", "RRD Apparels", "RR Garments", "SR Garments" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private double[,] GetDataSource()
{

```

```
double[,] dataSource = new double[,]
{
    { 52, 65, 67, 45, 37, 52, 32 },
    { 68, 52, 63, 51, 30, 51, 51 },
    { 60, 50, 42, 53, 66, 70, 41 },
    { 66, 64, 46, 40, 47, 41, 45 },
    { 65, 42, 58, 32, 36, 44, 49 },
    { 54, 46, 61, 46, 40, 39, 41 },
    { 48, 46, 61, 47, 49, 41, 41 },
    { 69, 52, 41, 44, 41, 52, 46 },
    { 50, 59, 44, 43, 27, 42, 26 },
    { 47, 49, 66, 53, 50, 34, 31 },
    { 61, 40, 62, 26, 34, 54, 56 }
};
return dataSource;
}
```

Providing line breaks

Axis labels with line breaks improve the readability of the HeatMap by splitting the text on an axis into multiple lines. The “
” character is used to add line breaks to the axis labels.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").XAxis(
    xAxis =>
xAxis.Labels(ViewBag.xLabels).OpposedPosition(true)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels).MaxLabelLength(50)).DataSource(ViewBag.dataSource).Render()
```

LINE-BREAKS.CS

```
public ActionResult TextStyle()
{
    string[] xlabels = new string[] { "Actual<br>Accept", "Actual<br>Reject" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[] { "Actual<br>Accept", "Actual<br>Reject" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private double[,] GetDataSource()
{
    double[,] dataSource = new double[,]
    {
        { 1, 76 },
        { 19, 3 }
    };
    return dataSource;
}
```

Customizing labels when intersecting with other labels

When the axis labels intersect, [LabelIntersectAction](#) property is used to handle the intersection. The [LabelIntersectAction](#) property can take the following values.

- **None** - It specifies that no action is taken when the axis labels intersect.
- **Trim** - It specifies to trim the axis labels when they intersect.
- **Rotate45** - When the axis labels intersect, they are rotated to 45 degrees.
- **MultipleRows** - It specifies to show all the axis labels as multiple rows when they intersect.

The below example demonstrates to trim the axis labels by using the [LabelIntersectAction](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Product wise
Monthly sales revenue for a e-commerce website")).
    TextStyle(ViewBag.textStyle)).XAxis(
    xAxis =>
xAxis.Labels(ViewBag.xLabels).OpposedPosition(true).LabelIntersectAction(LabelIntersectAction.Trim).EnableTrim(true)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels).LabelIntersectAction(LabelIntersectAction.Trim).EnableTrim(true)).PaletteSettings(ps => ps.Palette(palette => {
    palette.Color("#F0C27B").Add();
    palette.Color("#4B1248").Add();
})).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()
```

LABELINTERSECTACTION.CS

```
public ActionResult LabelIntersectAction()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[] { "Month of Feburary 2023", "Month of
March 2023", "Month of April 2023", "Month of May 2023", "Month of June
2023", "Month of July 2023", "Month of August 2023", "Month of September
2023", "Month of October 2023", "Month of November 2023", "Month of December
2023" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[] { "Ace Apparels", "Alpha Apparels", "RL
Garments", "RRD Garments", "RRD Apparels", "RR Garments", "SR Garments" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private double[,] GetDataSource()
{
    double[,] dataSource = new double[, ]
    {
```



```

        { 52, 65, 67, 45, 37, 52, 32 },
        { 68, 52, 63, 51, 30, 51, 51 },
        { 60, 50, 42, 53, 66, 70, 41 },
        { 66, 64, 46, 40, 47, 41, 45 },
        { 65, 42, 58, 32, 36, 44, 49 },
        { 54, 46, 61, 46, 40, 39, 41 },
        { 48, 46, 61, 47, 49, 41, 41 },
        { 69, 52, 41, 44, 41, 52, 46 },
        { 50, 59, 44, 43, 27, 42, 26 },
        { 47, 49, 66, 53, 50, 34, 31 },
        { 61, 40, 62, 26, 34, 54, 56 }
    };
    return dataSource;
}

```

Rotating labels

The axis labels can be rotated to the desired angles by using the [LabelRotation](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts=> ts.Text("Product wise
Monthly sales revenue for a e-commerce website").
    TextStyle(ViewBag.textStyle)).XAxis(
    xAxis =>
xAxis.Labels(ViewBag.xLabels).OpposedPosition(true).LabelRotation(90)).YAxis
(yaxis =>

yaxis.Labels(ViewBag.yLabels).LabelRotation(45)).PaletteSettings(ps =>
ps.Palette(palette => {
    palette.Color("#F0C27B").Add();
    palette.Color("#4B1248").Add();
})).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()

```

LABELROTATION.CS

```

public ActionResult LabelRotation()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[] { "Month of Feburary 2023", "Month of
March 2023", "Month of April 2023", "Month of May 2023", "Month of June
2023", "Month of July 2023", "Month of August 2023", "Month of September
2023", "Month of October 2023", "Month of November 2023", "Month of December
2023" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[] { "Ace Apparels", "Alpha Apparels", "RL
Garments", "RRD Garments", "RRD Apparels", "RR Garments", "SR Garments" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
}

```

```

    return View();
}
private double[,] GetDataSource()
{
    double[,] dataSource = new double[, ]
    {
        { 52, 65, 67, 45, 37, 52, 32 },
        { 68, 52, 63, 51, 30, 51, 51 },
        { 60, 50, 42, 53, 66, 70, 41 },
        { 66, 64, 46, 40, 47, 41, 45 },
        { 65, 42, 58, 32, 36, 44, 49 },
        { 54, 46, 61, 46, 40, 39, 41 },
        { 48, 46, 61, 47, 49, 41, 41 },
        { 69, 52, 41, 44, 41, 52, 46 },
        { 50, 59, 44, 43, 27, 42, 26 },
        { 47, 49, 66, 53, 50, 34, 31 },
        { 61, 40, 62, 26, 34, 54, 56 }
    };
    return dataSource;
}

```

Label formatting

HeatMap supports formatting the axis labels by using the [LabelFormat](#) property. Using this property, you can customize the axis label by global string format ('P', 'C', etc) or customized format like '{value}°C'.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Monthly Flight Traffic at JFK
Airport").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1,
1)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years).LabelFormat("yyy
y");
}).YAxis(yaxis =>
yaxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.Numeric).LabelFormat("${val
ue}").LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()

```

LABEL-FORMAT.CS

```

public ActionResult LabelFormat()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    ViewBag.dataSource = GetDataSource();
    return View();
}

```

```

}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

Axis intervals

In HeatMap, you can define an interval between the axis labels using the [Interval](#) property. In date-time axis, you can change the interval mode by using the [IntervalType](#) property. The date-time axis supports the following interval types:

- Years
- Months
- Days
- Hours
- Minutes

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Monthly Flight Traffic at JFK
Airport").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
{
    xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.DateTime).Minimum(new
System.DateTime(2007, 1, 1))
    .Maximum(new System.DateTime(2017, 1,
1)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Years)
    .LabelFormat("yyyy").Interval(2);
}).YAxis(yaxis => yaxis.Labels( ViewBag.yLabels)).CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(false).Format("{value}
flights")).LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Render()

```

INTERVAL.CS

```

public ActionResult Interval()
{

```

```

ViewBag.textStyle = new
{
    size= "15px",
    fontWeight= "500",
    fontStyle= "Normal",
    fontFamily= "Segoe UI"
};
string[] yLabels = new string[12] { "Jan", "Feb", "Mar", "Apr",
"May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"};
ViewBag.yLabels = yLabels;
ViewBag.border = new { width = 0};
ViewBag.dataSource = GetDataSource();
return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964,
36543, 30529, 33298, 36985},
            {34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467,
35390, 34196, 35302, 35703},
            {34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255,
32776, 32645, 31539, 32981},
            {32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837,
32910, 33437, 30659, 31965},
            {31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497,
34285, 34094, 32256, 33699},
            {31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943,
34128, 30624, 32398, 33522},
            {32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301,
33627, 34115, 31072, 33939},
            {32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978,
35777, 35277, 31281, 35411},
            {32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335,
37177, 37443, 32457, 37304},
            {34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868,
38252, 36505, 29576, 36450},
            {35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002,
37021, 36583, 32408, 37108}
        }
    };
    return data;
}

```

Axis label increment

Axis label increment in the HeatMap is used to display the axis labels with regular interval values in numeric and date-time axes. The labels will be displayed with tick gaps when you set the label interval. But, to achieve the same behavior without tick gaps, use the label increment. You can set the axis label increment using the [Increment](#) property and the default value of this property is 1.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").XAxis(xAxis =>

```

```
xAxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.Numeric).Minimum(0).Increment(2)).YAxis(yaxis =>
yaxis.ValueType(Syncfusion.EJ2.HeatMap.ValueType.Numeric).Minimum(0).Increment(3)).DataSource(ViewBag.dataSource).Render()
```

INCREMENT.CS

```
public ActionResult Increment()
{
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
        {74, 33, 88, 23, 86, 59}
    };
    return data;
}
```

Limiting labels in date-time axis

You can display the axis labels at specific time intervals along with the date-time axis using the [ShowLabelOn](#) property. This property supports the following types:

- **None**: Displays the axis labels based on the `IntervalType` and `Interval` property of the axis. This type is default value of the `ShowLabelOn` property.
- **Years**: Displays the axis labels on every year between given date-time range.
- **Months**: Displays the axis labels on every month between given date-time range.
- **Days**: Displays the axis labels on every day between given date-time range.
- **Minutes**: Displays the axis labels on every minute between given date-time range.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Summary of
merge requests in GitLab").TextStyle(ViewBag.textStyle)).XAxis(
xAxis =>
{
    xAxis.Minimum(new System.DateTime(2017, 6, 23)).Maximum(new
System.DateTime(2018, 6,
```

```

30)).IntervalType(Syncfusion.EJ2.HeatMap.IntervalType.Days).ValueType(Syncfu
sion.EJ2.HeatMap.ValueType.DateTime).
    ShowLabelOn(Syncfusion.EJ2.HeatMap.LabelType.Months).
    LabelFormat("MMM").OpposedPosition(true).Increment(7);
}).YAxis(yaxis =>
{
    yaxis.Labels(ViewBag.yLabels).IsInversed(true);
}).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Bottom).Width("20%").ShowL
abel(true).
Alignment(Syncfusion.EJ2.HeatMap.Alignment.Near).LabelDisplayType(Syncfusion
.EJ2.HeatMap.LabelDisplayType.None).
EnableSmartLegend(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("rgb(238,238,238)").Label("no
contributions").Add();
    palette.Value(1).Color("rgb(172, 213, 242)").Label("1-15
contributions").Add();
    palette.Value(16).Color("rgb(127, 168, 201)").Label("16-31
contributions").Add();
    palette.Value(32).Color("rgb(82, 123, 160)").Label("31-49
contributions").Add();
    palette.Value(50).Color("rgb(37, 78, 119)").Label("50+
contributions").Add();
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed).EmptyPointColor("white")).
Height("300px").CellSettings(cs =>
cs.Border(ViewBag.border).ShowLabel(false)).TooltipRender("tooltipRender").D
ataSource(ViewBag.dataSource).Render()
<script>
    var tooltipRender = function (args) {
        var intl = new ej.base.Internationalization();
        var format = intl.getDateFormat({ format: 'EEE MMM dd, yyyy' });
        var newDate = args.xValue;
        var date = new Date(newDate.getTime());
        var axisLabel = args.heatmap.axisCollections[1].axisLabels;
        var index = axisLabel.indexOf(args.yLabel);
        (date).setDate((date).getDate() + index);
        var value = format(date);
        args.content = [(args.value === 0 ? 'No' : args.value) + ' ' +
'contributions' + '<br>' + value];
    };
</script>

```

SHOW-LABEL.CS

```

public ActionResult ShowLabel()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    string[] yLabels = new string[7] { "Sun", "Mon", "Tue", "Wed", "Thu",
"Fri", "Sat" };

```

```

ViewBag.yLabels = yLabels;
ViewBag.border = new { color = "white" };
ViewBag.dataSource = GetDataSource();
return View();
}
private int[,] GetDataSource()
{
    int?[,] dataSource = new int?[,]
    {
        { null, null, null, null, 16, 48, 0 },
        { 0, 15, 0, 24, 0, 39, 0 },
        { 0, 18, 37, 0, 0, 50, 0 },
        { 0, 10, 0, 0, 44, 5, 0 },
        { 0, 36, 0, 45, 20, 18, 0 },
        { 0, 28, 1, 42, 0, 10, 0 },
        { 0, 16, 32, 0, 1, 25, 0 },
        { 0, 31, 2, 9, 24, 0, 0 },
        { 0, 8, 47, 0, 0, 35, 0 },
        { 0, 31, 0, 0, 0, 40, 0 },
        { 0, 8, 0, 27, 0, 35, 0 },
        { 0, 12, 9, 45, 0, 8, 0 },
        { 0, 0, 13, 0, 22, 10, 0 },
        { 0, 16, 32, 0, 1, 25, 0 },
        { 0, 31, 2, 9, 24, 0, 0 },
        { 0, 8, 47, 27, 0, 35, 0 },
        { 0, 28, 14, 10, 0, 0, 0 },
        { 0, 36, 0, 45, 20, 18, 0 },
        { 0, 28, 1, 42, 0, 10, 0 },
        { 0, 31, 0, 24, 0, 40, 0 },
        { 0, 8, 47, 27, 0, 35, 0 },
        { 0, 36, 0, 45, 20, 18, 0 },
        { 0, 28, 1, 42, 0, 10, 0 },
        { 0, 31, 0, 24, 0, 40, 0 },
        { 0, 16, 32, 0, 1, 25, 0 },
        { 0, 31, 2, 9, 24, 0, 0 },
        { 0, 8, 47, 27, 0, 35, 0 },
        { 0, 10, 0, 36, 23, 19, 0 },
        { 0, 18, 37, 23, 0, 50, 0 },
        { 0, 28, 14, 10, 0, 0, 0 },
        { 0, 18, 37, 23, 0, 50, 0 },
        { 0, 18, 37, 23, 0, 50, 0 },
        { 0, 28, 14, 10, 0, 0, 0 },
        { 0, 31, 2, 9, 24, 0, 0 },
        { 0, 8, 47, 27, 0, 35, 0 },
        { 0, 10, 2, 0, 44, 5, 0 },
        { 0, 36, 0, 45, 20, 18, 0 },
        { 0, 28, 1, 42, 0, 10, 0 },
        { 0, 31, 0, 24, 0, 40, 1 },
        { 0, 16, 32, 0, 1, 25, 0 },
        { 0, 31, 2, 9, 24, 0, 0 },
        { 0, 8, 47, 27, 0, 35, 0 },
        { 0, 10, 2, 0, 44, 5, 0 },
        { 0, 12, 9, 45, 0, 8, 0 },
        { 0, 0, 13, 35, 22, 10, 0 },
        { 0, 28, 14, 10, 0, 0, 0 },
        { 0, 36, 0, 45, 20, 18, 2 },
        { 0, 28, 1, 42, 0, 10, 0 },
    }
}

```

```

        {0, 31, 0, 24, 0, 40, 1},
        {0, 8, 47, 27, 0, 35, 0},
        {0, 10, 2, 0, 44, 5, 0},
        {0, 31, 2, 9, 24, 0, 1},
        {0, 8, 47, 27, 0, 35, 40},
        {0, 10, 2, 0, 44, 5, null},
    };
    return data;
}

```

Multilevel Labels

Multilevel labels are used to classify a group of axis labels as a single category, which is then displayed with a label. By using [MultiLevelLabels](#) tag, you can add multiple levels on top of the axis labels.

To divide and group the axis labels, you can use [MultiLevelLabels](#) property. The starting and ending indexes of the axis labels can be set using the [Start](#) and [End](#) properties in the [Categories](#). The [Text](#) property can be used to specify a name for the grouped axis labels.

The multilevel labels can be customized by using the following properties and tags.

- [Overflow](#) - It is used to trim or wrap the multilevel labels when the label overflows the intended space. NOTE: This property is only for x-axis.
- [Alignment](#) - It is used to place and align the multilevel labels.
- [MaximumTextWidth](#) - It is used to set the maximum width of the text. When the text length exceeds the maximum text width, the overflow action will be performed.
- [TextStyle](#) - It is used to customize the font style of the multilevel labels.
- [Border](#) - It is used to customize the border of the multilevel labels displayed in the x-axis and y-axis.

CSHTML

```

@using Syncfusion.EJ2;
@{
    var multilevelXAxisBorder = new
Syncfusion.EJ2.HeatMap.HeatMapAxisLabelBorder
    {
        Color = "#a19d9d",
        Width = 1,
        Type = BorderType.Rectangle
    };
    var multilevelYAxisBorder = new
Syncfusion.EJ2.HeatMap.HeatMapAxisLabelBorder
    {
        Color = "#a19d9d",
        Width = 1,
        Type = BorderType.Brace
    };
}
@Html.EJS().HeatMap("container").Width("700px").Height("500px").TitleSetting
s(ts => ts.Text("Product wise Monthly sales revenue for a e-commerce
website")).
    TextStyle(ViewBag.textStyle)).XAxis(

```



```

        xAxis => xAxis.Border(border =>
border.Color("#a19d9d").Width(1).Type(BorderType.Rectangle)).MultiLevelLabels(multilevel => {
    multilevel.Overflow(TextOverflow.Trim)

.Alignment(Alignment.Near).Border(multilevelXAxisBorder).Add();
})
    .Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.MultiLevelLabels(multilevel => {
        multilevel.Border(multilevelYAxisBorder).Add();
        multilevel.Border(multilevelYAxisBorder).Add();
        multilevel.Border(multilevelYAxisBorder).Add();
    }).Labels(ViewBag.yLabels)).PaletteSettings(ps =>
ps.Palette(palette => {
    palette.Color("#F0C27B").Add();
    palette.Color("#4B1248").Add();
})) .LegendSettings(ls =>
ls.Visible(false)).DataSource(ViewBag.dataSource).Load("load").Render()
<script>
    var load = function (args) {
        args.heatmap.xAxis.multiLevelLabels[0].categories = [
            { start: 0, end: 2, text: 'Electronics' },
            { start: 3, end: 4, text: 'Beauty and personal care' },
            { start: 5, end: 7, text: 'Fashion' },
            { start: 8, end: 10, text: 'Household' }
        ];
        args.heatmap.yAxis.multiLevelLabels[0].categories = [
            {start: 0, end: 2, text: 'Q1' },
            {start: 3, end: 5, text: 'Q2' },
            {start: 6, end: 8, text: 'Q3' },
            {start: 9, end: 11, text: 'Q4' }
        ];
        args.heatmap.yAxis.multiLevelLabels[1].categories = [
            {start: 0, end: 5, text: 'First Half Yearly' },
            {start: 6, end: 11, text: 'Second Half Yearly' }
        ];
        args.heatmap.yAxis.multiLevelLabels[2].categories = [
            {start: 0, end: 11, text: 'Yearly' }
        ];
    };
</script>

```

MULTI-LEVEL-LABELS.CS

```

public ActionResult MultiLevelLabel()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[] { "Laptop", "Mobile", "Gaming",
    "Cosmetics", "Fragrance", "Watches", "Handbags", "Apparels", "Kitchenware",
    "Furniture", "Home Decor" };
}

```

```

ViewBag.xLabels = xlabels;
string[] yLabels = new string[] { "Jan", "Feb", "Mar", "Apr", "May",
"Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
ViewBag.yLabels = yLabels;
ViewBag.dataSource = GetDataSource();
return View();
}
private double[,] GetDataSource()
{
    double[,] dataSource = new double[, ]
    {
        { 52, 65, 67, 45, 37, 52, 32, 76, 60, 64, 82, 91 },
        { 68, 52, 63, 51, 30, 51, 51, 81, 70, 60, 88, 80 },
        { 60, 50, 42, 53, 66, 70, 41, 69, 76, 74, 86, 97 },
        { 66, 64, 46, 40, 47, 41, 45, 76, 83, 69, 92, 84 },
        { 65, 42, 58, 32, 36, 44, 49, 79, 83, 69, 83, 93 },
        { 54, 46, 61, 46, 40, 39, 41, 69, 61, 84, 84, 87 },
        { 48, 46, 61, 47, 49, 41, 41, 67, 78, 83, 98, 87 },
        { 69, 52, 41, 44, 41, 52, 46, 71, 63, 84, 83, 91 },
        { 50, 59, 44, 43, 27, 42, 26, 64, 76, 65, 81, 86 },
        { 47, 49, 66, 53, 50, 34, 31, 79, 78, 79, 89, 95 },
        { 61, 40, 62, 26, 34, 54, 56, 74, 83, 78, 95, 98 }
    };
    return dataSource;
}

```

Palette in ASP.NET MVC HeatMap Chart Component

In heat map, each data point is displayed as a cell with applied color based on the data value. The palette in the heat map is used to define the color range for cells and gradient type for colors. You can define the colors either in RGB or hex codes using the [color](#) property in the [palette](#). The defined colors are applied to the cell background based on the palette type and cell value.

Palette types

You can display the heat map cells either in gradient colors or fixed colors.

Gradient

The smooth transition between the given palette colors can be applied for the heat map cells based on value. The heat map calculates all the gradient colors between the start and end colors for all distinct data values. Default start color and end color will be considered for gradient calculation, if the colors are not defined. The palette type must be defined as **Gradient** for the [type](#) property in the [paletteSettings](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
}

```

```
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).DataSource(ViewBag.dataSource).Render()
```

GRADIENT.CS

```
public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Fixed

In fixed palette type, solid colors are applied to the heat map cells. The data values can be grouped based on the number of colors defined for the heat map. The palette type should be defined as **Fixed** for the [type](#) property in the `paletteSettings` property.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.HeatMap;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
```

```
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).LegendSettings(ls =>
ls.Position(LegendPosition.Left)).PaletteSettings(ps =>
ps.Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed)).DataSource(ViewBag.dataSo
urce).Render()
```

FIXED.CS

```
public ActionResult Fixed()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Defining color stops

You can define the colors ranges or color stops for data values in both gradient and fixed palette types. You need to define the data value in the [value](#) property for [palette](#) property to calculate the color stops. The heat map automatically calculates the color stops if the [value](#) property is not defined. The [label](#)

property is used to provide the additional information about the color that is to be displayed in the legend. If the label is not provided, the value is displayed in the legend. The labels can be automatically calculated based on data values, if both the values and labels are not defined.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Label("Low").Value(50).Add();
    palette.Color("#6C5B7B").Label("Moderate").Value(80).Add();
    palette.Color("#355C7D").Label("High").Value(100).Add();
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).DataSource(ViewBag.dataSource).Render()
```

COLOR.CS

```
public ActionResult PaletteColor()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
    }
}
```

```

        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

See Also

- [How to enable smart legend](#)

Legend in ASP.NET MVC HeatMap Chart Component

The legend is used to provide the information about the heat map cell. You can enable the legend by setting the [visible](#) property to **true**.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("#C2E7EC").Add();
    palette.Value(10).Color("#AEDFE6").Add();
    palette.Value(20).Color("#9AD7E0").Add();
    palette.Value(30).Color("#72C7D4").Add();
    palette.Value(40).Color("#5EBFCE").Add();
    palette.Value(50).Color("#4AB7C8").Add();
    palette.Value(60).Color("#309DAE").Add();
    palette.Value(70).Color("#2B8C9B").Add();
    palette.Value(80).Color("#206974").Add();
    palette.Value(90).Color("#15464D").Add();
    palette.Value(100).Color("#000000").Add();
})) .LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right)).DataSource(ViewBag
.dataSource).Render()

```

DEFAULT.CS

```

public ActionResult Legend()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
}

```

```

        ViewBag.yLabels = yLabels;
        ViewBag.dataSource = GetDataSource();
        return View();
    }
    private int[,] GetDataSource()
    {
        int[,] data = new int[,]
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        };
        return data;
    }
}

```

Legend types

Heat map supports two legend types: Gradient and list type.

- **Gradient** - This is a continuous color legend with smooth color transition between palette color values.
- **List** - List is a fixed color legend. Each palette color information is shown separately in the list item.

You can change the legend type by using the [type](#) property in the `paletteSettings` property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).LegendSettings(ls =>
ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Color("#C06C84").Add();
    palette.Color("#6C5B7B").Add();
    palette.Color("#355C7D").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed)).DataSource(ViewBag.dataSource).Render()

```

TYPES.CS

```

public ActionResult LegendTypes()

```

```

{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    ViewBag.textStyle = new { fontStyle = "Italic",fontFamily = "Segoe
    UI" };
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

Placement

You can place the legend at left, right, top, or bottom to the heat map layout by using the [position](#) property. The legend is positioned at the right to the heat map by default.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Top)).DataSource(ViewBag.d
ataSource).Render()

```


PLACEMENT.CS

```

public ActionResult Placement()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}

```

Alignment

You can align the legend as center, far, or near to the heat map using the [alignment](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Top).Alignment(Syncfusion.
EJ2.HeatMap.Alignment.Near).Width("150px")).DataSource(ViewBag.dataSource).R
ender()

```

ALIGNMENT.CS

```

public ActionResult Alignment()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}

```

Legend dimensions

You can change the legend dimensions with values in pixels or percentage by using the [width](#) and [height](#) properties.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right).Height("150px")).Da
taSource(ViewBag.dataSource).Render()

```

DIMENSIONS.CS

```

public ActionResult Dimensions()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}

```

Paging for legend

Paging is available only for the list type legend in the heat map, and it can be enabled by default, when the legend items exceed the legend bounds. You can view each legend items by navigating between the pages using navigation buttons.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("#C2E7EC").Add();
}

```

```

palette.Value(10).Color("#AEDFE6").Add();
palette.Value(20).Color("#9AD7E0").Add();
palette.Value(25).Color("#86CFDA").Add();
palette.Value(30).Color("#72C7D4").Add();
palette.Value(40).Color("#5EBFCE").Add();
palette.Value(50).Color("#4AB7C8").Add();
palette.Value(55).Color("#36AFC2").Add();
palette.Value(60).Color("#309DAE").Add();
palette.Value(70).Color("#2B8C9B").Add();
palette.Value(75).Color("#257A87").Add();
palette.Value(80).Color("#206974").Add();
palette.Value(85).Color("#1B5761").Add();
palette.Value(90).Color("#15464D").Add();
palette.Value(100).Color("#000000").Add();
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right).Height("150px")).DataSource(ViewBag.dataSource).Render()

```

PAGING.CS

```

public ActionResult Paging()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
}

```

```

        return data;
    }

```

Smart Legend

Smart legend is another way of showing list type legend with responsiveness and readability, when the palette has more number of items. You can enable this smart legend by using the [enableSmartLegend](#) property when the palette type is set to **Fixed**.

In smart legend, you can change the display type of legend labels by using the [labelDisplayType](#) property.

The following are the legend label display types:

- **All**: Displays all labels in the legend.
- **Edge**: Displays the legend labels only at extreme ends.
- **None**: None of the labels are displayed. The tooltip will appear for this type of label display when hovering over the legend item.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("#C2E7EC").Add();
    palette.Value(10).Color("#AEDFE6").Add();
    palette.Value(20).Color("#9AD7E0").Add();
    palette.Value(30).Color("#72C7D4").Add();
    palette.Value(40).Color("#5EBFCE").Add();
    palette.Value(50).Color("#4AB7C8").Add();
    palette.Value(60).Color("#309DAE").Add();
    palette.Value(70).Color("#2B8C9B").Add();
    palette.Value(80).Color("#206974").Add();
    palette.Value(90).Color("#15464D").Add();
    palette.Value(100).Color("#000000").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed) .LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Bottom) .Width("75%") .Enabl
eSmartLegend(true)) .DataSource(ViewBag.dataSource) .Render()

```

SMART-LEGEND.CS

```

public ActionResult Legend()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
}

```

```

        string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
        "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
        "Mario" };
        ViewBag.xLabels = xlabels;
        string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
        "Fri", "Sat" };
        ViewBag.yLabels = yLabels;
        ViewBag.dataSource = GetDataSource();
        return View();
    }
    private int[,] GetDataSource()
    {
        int[,] data = new int[,]{
            {
                {73, 39, 26, 39, 94, 0},
                {93, 58, 53, 38, 26, 68},
                {99, 28, 22, 4, 66, 90},
                {14, 26, 97, 69, 69, 3},
                {7, 46, 47, 47, 88, 6},
                {41, 55, 73, 23, 3, 79},
                {56, 69, 21, 86, 3, 33},
                {45, 7, 53, 81, 95, 79},
                {60, 77, 74, 68, 88, 51},
                {25, 25, 10, 12, 78, 14},
                {25, 56, 55, 58, 12, 82},
                {74, 33, 88, 23, 86, 59}
            }
        };
        return data;
    }
}

```

Legend Selection

In the HeatMap, the legend selection is used to toggle the visibility of cell for viewing the specific range value. You can enable the legend selection using the [toggleVisibility](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("#C2E7EC").Add();
    palette.Value(10).Color("#AEDFE6").Add();
    palette.Value(20).Color("#9AD7E0").Add();
    palette.Value(30).Color("#72C7D4").Add();
    palette.Value(40).Color("#5EBFCE").Add();
    palette.Value(50).Color("#4AB7C8").Add();
    palette.Value(60).Color("#309DAE").Add();
    palette.Value(70).Color("#2B8C9B").Add();
    palette.Value(80).Color("#206974").Add();
    palette.Value(90).Color("#15464D").Add();
    palette.Value(100).Color("#000000").Add();
})) .Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed).LegendSettings(ls =>

```

```
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Bottom).Width("75%").EnableSmartLegend(true).ToggleVisibility(true)).DataSource(ViewBag.dataSource).Render()
```

LEGEND-SELECTION.CS

```
public ActionResult Legend()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };

    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}

private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Legend Title

The legend title displays a specific information about the legend. You can enable the legend title by setting the [title](#) property by providing the text and customizing the legend title text style using the [textStyle](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee")).XAxis(xAxis =>
```

```

xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("#6EB5D0").Add();
    palette.Value(50).Color("#7EDCA2").Add();
    palette.Value(100).Color("#DCD57E").Add();
}).Type(Syncfusion.EJ2.HeatMap.PaletteType.Fixed)).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Bottom).Width("75%").Enabl
eSmartLegend(true).ToggleVisibility(true).Title(title=>title.Text("1000
US$"))).DataSource(ViewBag.dataSource).Render()

```

LEGEND-TITLE.CS

```

public ActionResult Legend()
{
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}

```

Appearance in ASP.NET MVC HeatMap Chart Component

Cell customization

Border

Change the width, color, and radius of the heat map cells by using the [Border](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>

```



```
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.Border(ViewBag.border)).DataSource(ViewBag.dataSource).Render()
```

BORDER.CS

```
public ActionResult Border()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    ViewBag.border = new { width = 1, radius = 4, color= "white" };
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}
```

Cell highlighting

Enable or disable the cell highlighting while hovering over the heat map cells by using the [EnableCellHighlighting](#) property.

Note: The cell highlighting only works in a SVG rendering mode.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.EnableCellHighlighting(true)).DataSource(ViewBag.dataSource).Render()
```

CELL-HIGHLIGHT.CS

```
public ActionResult CellHighlight()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Color gradient mode

The [ColorGradientMode](#) property can be used to set the minimum and maximum values for colors based on row and column. Three types of color gradient modes are available.

- **Table:** The minimum and maximum value colors calculated for overall data.

- **Row:** The minimum and maximum value colors calculated for each row of data.
- **Column:** The minimum and maximum value colors calculated for each column of data.

Note: The default value of `ColorGradientMode` is `Table`.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.EnableCellHighlighting(true)).PaletteSettings(ps =>
ps.ColorGradientMode(Syncfusion.EJ2.HeatMap.ColorGradientMode.Column)).DataS
ource(ViewBag.dataSource).Render()
```

COLORMODE.CS

```
public ActionResult CellHighlight()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

```
}

```

Background color

The background color of the HeatMap can be customized using the [BackgroundColor](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").BackgroundColor("#c7afcf").TitleSettings(ts
=>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
```

BACKGROUNDCOLOR.CS

```
public ActionResult BackgroundColor()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Margin

Set the margin for the heat map from its container by using the [Margin](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Margin(margin =>
margin.Left(15).Right(15).Top(15).Bottom(15)).TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
```

MARGIN.CS

```
public ActionResult Margin()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Title

The title is used to provide a quick information about the data plotted in heatmap. The [Text](#) property is used to set the title for the heatmap. The text style of the title can be customized by using the [TextStyle](#) property.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
```

TITLE.CS

```
public ActionResult Title()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Italic",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Data label

The visibility of data labels can be toggled using the [ShowLabel](#) property. By default, the data labels will be visible.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).DataSource(ViewBag.dataSource).Render()
```

DATA-LABEL.CS

```
public ActionResult DataLabel()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Customize the data label

The value displayed in the HeatMap cell can be changed using the [CellRender](#) event.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellRender("cellRender").DataSource(ViewBag.d
ataSource).Render()
<script>
var cellRender = function (args) {
    args.displayText = (args.value) + '$';
};
</script>
```

CELLRENDER.CS

```
public ActionResult CellRender()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
};
```



```

        return data;
    }

```

Text style

The text attributes of the data label such as font-family, font-size, and color can be customized using the [TextStyle](#) in the [CellSettings](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.TextStyle(ViewBag.cellTextStyle)).DataSource(ViewBag.dataSource).Render()

```

TEXT-STYLE.CS

```

public ActionResult TextStyle()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    ViewBag.cellTextStyle = new { fontStyle = "Italic",fontFamily =
    "Segoe UI" };
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
    }
}

```

```

        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

Format

The format of the data label, such as currency, decimal, percent etc. can be changed using [Format](#) property.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.Format("{value} $")).DataSource(ViewBag.dataSource).Render()

```

FORMAT.CS

```

public ActionResult Format()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
        }
    };
}

```

```

        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

Template

Array binding

By including `${value}` in the template content, the value from the data source for the corresponding cell can be displayed in the HeatMap cell as data label template content. Additionally, the x-axis and y-axis label values can be displayed by including `${xLabel}` and `${yLabel}` in the template content.

Table

The following example demonstrates how to add a data label template for array table binding.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2015 -
2017").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
    ls.Visible(true)).CellSettings(cs =>
    cs.LabelTemplate("<div
style='width:25px;height:20px;text-align:center;padding-top:2px;background-
color:#5BBB9C; border: 1px solid #000000; border-radius:50%;font-
weight:bold;'>${value}</div>")).DataSource("dataSource").Render()
}
<script>
    var dataSource = [
        [[4, 39], [3, 8], [1, 3], [1, 10], [4, 4], [2, 15]],
        [[4, 28], [5, 92], [5, 73], [3, 1], [3, 4], [4, 126]],
        [[4, 45], [5, 152], [0, 44], [4, 54], [5, 243], [2, 45]]
    ];
</script>

```

TEMPLATE-ARRAY-TABLE.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    string[] xlabels = new string[3] { "2015", "2016", "2017" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-
Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```

Cell

The following example demonstrates how to add a data label template for array cell binding.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2015 - 2017").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
    ls.Visible(true)).CellSettings(cs =>
    cs.LabelTemplate("<div
style='width:25px;height:20px;text-align:center;padding-top:2px;background-
color:#5BBB9C; border: 1px solid #000000; border-radius:50%;font-
weight:bold;'>${value}</div>")).DataSource("dataSource").DataSourceSettings(ds => {
ds.IsJsonData(false).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell);
}).Render()
<script>
    var dataSource = [
        [0, 0, [4, 39]], [0, 1, [3, 8]], [0, 2, [1, 3]], [0, 3, [1, 10]], [0, 4,
[4, 4]], [0, 5, [2, 15]],
        [1, 0, [4, 28]], [1, 1, [5, 92]], [1, 2, [5, 73]], [1, 3, [3, 1]], [1,
4, [3, 4]], [1, 5, [4, 126]],
        [2, 0, [4, 45]], [2, 1, [5, 152]], [2, 2, [0, 44]], [2, 3, [4, 54]], [2,
4, [5, 243]], [2, 5, [2, 45]]
    ];
</script>
```

TEMPLATE-ARRAY-CELL.CS

```
public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    string[] xlabels = new string[3] { "2015", "2016", "2017" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-
Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}
```

JSON binding

By including the desired field name in the template content, such as **\${value}**, the value from the data source for the corresponding cell can be displayed in the HeatMap cell as data label template content.

Table

The following example demonstrates how to add a data label template for JSON table binding.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2015 -
2017").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
    ls.Visible(true)).PaletteSettings(ps => ps.Palette(palette
=> {
        palette.Color("#C06C84").Add();
        palette.Color("#6C5B7B").Add();
        palette.Color("#355C7D").Add();
    })).Type(Syncfusion.EJ2.HeatMap.PaletteType.Gradient)).CellSettings(cs =>
    cs.LabelTemplate("<div><img
style='width:20px;height:20px;' src='${image}'/>
</div>")).DataSource("dataSource").DataSourceSettings(ds => {
ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Table).XD
ataMapping("Year"); }).Render()
<script>
    var dataSource = [
        {
            Year: '2017',
            image:
                'https://ej2.syncfusion.com/demos/src/circular-
gauge/images/golfball.png',
            'Jan-Feb': [4, 39],
            'Mar-Apr': [3, 8],
            'May-Jun': [1, 3],
            'Jul-Aug': [1, 10],
            'Sep-Oct': [4, 4],
            'Nov-Dec': [2, 15]
        },
        {
            Year: '2016',
            image:
                'https://ej2.syncfusion.com/demos/src/circular-
gauge/images/basketball.png',
            'Jan-Feb': [4, 28],
            'Mar-Apr': [5, 92],
            'May-Jun': [5, 73],
            'Jul-Aug': [3, 1],
            'Sep-Oct': [3, 4],
            'Nov-Dec': [4, 126]
        },
        {
            Year: '2015',
            image:
                'https://ej2.syncfusion.com/demos/src/circular-
gauge/images/football.png',
            'Jan-Feb': [4, 45],
            'Mar-Apr': [5, 152],
```

```

        'May-Jun': [0, 44],
        'Jul-Aug': [4, 54],
        'Sep-Oct': [5, 243],
        'Nov-Dec': [2, 45]
    },
    ];
</script>

```

TEMPLATE-JSON-TABLE.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    string[] xlabels = new string[3] { "2015", "2016", "2017" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```

Cell

The following example demonstrates how to add a data label template for JSON cell binding.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
    ts.Text("Commercial Aviation Accidents and Fatalities by year 2015 - 2017").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).LegendSettings(ls =>
                ls.Visible(true)).CellSettings(cs =>
                    cs.LabelTemplate("<div> Accidents - ${Accidents}</div>")).DataSource("dataSource").DataSourceSettings(ds => {
                        ds.IsJsonData(true).AdaptorType(Syncfusion.EJ2.HeatMap.AdaptorType.Cell).XDataMapping("Year").YDataMapping("Months").ValueMapping("Fatalities");
                    }).Render()
    )
<script>
var dataSource = [
    { Year: '2017', Months: 'Jan-Feb', Accidents: 4, Fatalities: 39 },
    { Year: '2017', Months: 'Mar-Apr', Accidents: 3, Fatalities: 8 },
    { Year: '2017', Months: 'May-Jun', Accidents: 1, Fatalities: 3 },
    { Year: '2017', Months: 'Jul-Aug', Accidents: 1, Fatalities: 10 },
    { Year: '2017', Months: 'Sep-Oct', Accidents: 4, Fatalities: 4 },
    { Year: '2017', Months: 'Nov-Dec', Accidents: 2, Fatalities: 15 },
    { Year: '2016', Months: 'Jan-Feb', Accidents: 4, Fatalities: 28 },
    { Year: '2016', Months: 'Mar-Apr', Accidents: 5, Fatalities: 92 },
    { Year: '2016', Months: 'May-Jun', Accidents: 5, Fatalities: 73 },

```

```

    { Year: '2016', Months: 'Jul-Aug', Accidents: 3, Fatalities: 1 },
    { Year: '2016', Months: 'Sep-Oct', Accidents: 3, Fatalities: 4 },
    { Year: '2016', Months: 'Nov-Dec', Accidents: 4, Fatalities: 126 },
    { Year: '2015', Months: 'Jan-Feb', Accidents: 4, Fatalities: 45 },
    { Year: '2015', Months: 'Mar-Apr', Accidents: 5, Fatalities: 152 },
    { Year: '2015', Months: 'May-Jun', Accidents: 0, Fatalities: 0 },
    { Year: '2015', Months: 'Jul-Aug', Accidents: 4, Fatalities: 54 },
    { Year: '2015', Months: 'Sep-Oct', Accidents: 5, Fatalities: 243 },
    { Year: '2015', Months: 'Nov-Dec', Accidents: 2, Fatalities: 45 },
  ];
</script>

```

TEMPLATE-JSON-CELL.CS

```

public ActionResult Gradient()
{
    ViewBag.textStyle = new
    {
        size = "15px",
        fontWeight = "500",
        fontStyle = "Normal",
        fontFamily = "Segoe UI"
    };
    string[] xlabels = new string[3] { "2015", "2016", "2017" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Jan-Feb", "Mar-Apr", "May-Jun", "Jul-Aug", "Sep-Oct", "Nov-Dec" };
    ViewBag.yLabels = yLabels;
    return View();
}

```

Dimensions in ASP.NET MVC HeatMap Chart Component

Size for container

Heat map can be rendered to its container size. You can set the size through inline or CSS.

```

`javascript
<div id='container'>
<div id='element' style="width:650px; height:350px;"></div>
</div>
`

```

Size for heat map

You can set the size of heat map directly by using the [width](#) and [height](#) properties.

In pixel

You can set the size for heat map in a pixel.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Width("650px").Height("350px").TitleSettings(ts =>

```

```
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(true)).DataSource(ViewBag.dataSource).Render()
```

PIXEL.CS

```
public ActionResult Pixel()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

In percentage

By setting value in percentage, heat map gets its dimension with respect to its container. For example, when the height is '50%', heat map rendered to half of the container height.

CSHTML

```
@using Syncfusion.EJ2;
```



```
@Html.EJS().HeatMap("container").Width("80%").Height("90%").TitleSettings(ts
=>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(true)).DataSource(ViewBag.dataSource).Render()
```

PERCENTAGE.CS

```
public ActionResult Percentage()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}
```

Tooltip in ASP.NET MVC HeatMap Chart Component

Tooltip is used to provide the details of the heatmap cell, and this can be displayed, while hovering the cursor over the cell or performing tap action in touch devices.

Default tooltip

You can enable the tooltip by setting the [showTooltip](#) property to **true**.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Crude Oil
Production of Non-OPEC Countries (in Million barrels per
day)").TextStyle(ViewBag.textStyle)).XAxis(
xAxis =>{ xAxis.Labels(ViewBag.xLabels)}).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).ShowTooltip(true).DataSource(ViewBag.dataSource).Render
()
```

TOOLTIP.CS

```
public ActionResult Tooltip()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[8] { "Canada", "China", "Egypt",
    "Mexico", "Norway", "Russia", "UK", "USA"};
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[11] { "2000", "2001", "2002",
    "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58,
0.56],
            [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32,
1.23],
            [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57,
2.74],
            [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64,
2.61],
            [3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06,
1.87],
            [3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79,
4.07],
            [5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35,
5.47],
            [6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]
        }
    };
    return data;
}
```

Tooltip template

In heatmap, you can customize the tooltip using the [tooltipRender](#) client side event.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Crude Oil
Production of Non-OPEC Countries (in Million barrels per
day)").TextStyle(ViewBag.textStyle)).XAxis(
xAxis =>{ xAxis.Labels(ViewBag.xLabels)}).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).ShowTooltip(true).TooltipRender("tooltipRender").DataSo
urce(ViewBag.dataSource).Render()
<script>
    var tooltipRender = function (args) {
        args.content = ['In ' + args.yLabel + ', the ' + args.xLabel + '
produced ' + args.value + ' million barrels per day'];
    };
</script>
```

TEMPLATE.CS

```
public ActionResult Tooltiptemplate()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[8] { "Canada", "China", "Egypt",
    "Mexico", "Norway", "Russia", "UK", "USA"};
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[11] { "2000", "2001", "2002",
    "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {
            [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58,
0.56],
            [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32,
1.23],
            [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57,
2.74],
            [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64,
2.61],
            [3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06,
1.87],
            [3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79,
4.07],
        }
    };
}
```

```

                    [5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35,
5.47],
                    [6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]
                };
                return data;
            }

```

Customize the appearance of Tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Crude Oil
Production of Non-OPEC Countries (in Million barrels per
day)").TextStyle(ViewBag.textStyle)).XAxis(
xAxis =>{ xAxis.Labels(ViewBag.xLabels)}).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).ShowTooltip(true).TooltipRender("tooltipRender").Palett
eSettings(ps => ps.Palette(palette =>
{
    palette.Color("#F0ADCE").Add();
    palette.Color("#19307B").Add();
})).TooltipSettings(ts =>
    ts.Fill("#265259").Border(br =>
br.Color("#98BABF").Width(1)).TextStyle(style =>
style.Color("#FFFFFF").Size("12px"))).DataSource(ViewBag.dataSource).Render(
)
<script>
    var tooltipRender = function (args) {
        args.content = ['In ' + args.yLabel + ', the ' + args.xLabel + '
produced ' + args.value + ' million barrels per day'];
    };
</script>

```

TEMPLATE.CS

```

public ActionResult Tooltipfill()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };

    string[] xlabels = new string[8] { "Canada", "China", "Egypt",
    "Mexico", "Norway", "Russia", "UK", "USA"};
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[11] { "2000", "2001", "2002",
    "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
}

```

```
        return View();
    }
    private int[,] GetDataSource()
    {
        int[,] data = new int[,]
        {
            [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58,
0.56],
            [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32,
1.23],
            [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57,
2.74],
            [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64,
2.61],
            [3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06,
1.87],
            [3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79,
4.07],
            [5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35,
5.47],
            [6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]
        };
        return data;
    }
}
```

Selection in ASP.NET MVC HeatMap Chart Component

In the HeatMap, the cell selection is used to select single or multiple HeatMap cells at runtime and get the selected cell details using the [CellSelected](#) event. You can enable the cell selection using the [AllowSelection](#) property.

The HeatMap cells can be selected using the following interactions, as shown in the table below.

Modes of Interactions	Description
Mouse	HeatMap cells can be selected by clicking or dragging and dropping over them.
Touch	HeatMap cells can be selected by tapping or dragging and dropping over them.
Keyboard	The Ctrl key on the keyboard can be used to enable multiple cell selection with mouse and touch interaction. The Ctrl key can only be used if the EnableMultiSelect property is set to true in order to enable multiple cell selection.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
```

```
yaxis.Labels(ViewBag.yLabels)).AllowSelection(true).DataSource(ViewBag.dataSource).Render()
```

CELLSELECTION.CS

```
public ActionResult CellRender()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Enable single cell selection

In the HeatMap, the [EnableMultiSelect](#) property is used to allow single cell selection. When you set the [EnableMultiSelect](#) property to **false**, only one cell is selected. By default, [EnableMultiSelect](#) property is set to **true**.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
```

```
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).AllowSelection(true).EnableMultiSelect(false)
.DataSource(ViewBag.dataSource).Render()
```

SINGLECELLSELECTION.CS

```
public ActionResult SingleCellSelection()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

Clearing cell selection

The `clearSelection` method can be used to clear all the selected cells. The below example illustrates the same.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
```

```

xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).AllowSelection(true).DataSource(ViewBag.dataS
ource).Render()
@Html.EJS().Button("selection").IconCss("e-icons e-play-
icon").Content("ClearSelection").CssClass("e-flat").Render()
<script>
    document.getElementById('selection').onclick = function () {
        var heatmapObj =
document.getElementById('container').ej2_instances[0];
        heatmapObj.clearSelection();
    };
</script>

```

CLEARSELECTION.CS

```

public ActionResult ClearSelection()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}

```


Accessibility in ASP.NET MVC HeatMap chart component

HeatMap has built-in accessibility features like screen reading. Screen reading in the HeatMap component allows all users, regardless of ability or disability, to use the component. The following HeatMap elements will be read aloud with screen reading software like Narrator for Windows.

Elements	Description
--- ---	
Title	Reads the contents of the HeatMap chart's title.
Axis labels	Reads the x and y axis labels of the HeatMap chart.
Multilevel labels	Reads the multilevel labels in the x and y axis of the HeatMap chart.
Cell labels	Reads the labels from the cells in the Heatmap chart.
Legend title	Reads the contents of the legend's title as specified in HeatMap chart.
Legend item label	Reads the label of a legend item in HeatMap chart.

Ensuring accessibility

The HeatMap component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the HeatMap component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the HeatMap component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Events in ASP.NET MVC HeatMap Chart Component

This section describes the HeatMap chart event, which occurs when the required actions are performed.

CellClick

When you click on a HeatMap cell, the [CellClick](#) event is triggered.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").CellClick("heatmapCellClick").TitleSettings
(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
function heatmapCellClick(args) {
    console.log("The cell click event has been triggered!!!");
}
</script>
```

CELLCLICK.CS

```
public ActionResult CellClick()
```

```

{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

CellDoubleClick

When you double click on a HeatMap cell, the [CellDoubleClick](#) event is triggered.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").CellDoubleClick("heatmapCellDoubleClick").TitleSettings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapCellDoubleClick(args) {
        console.log("The cell double click event has been triggered!!!");
    }
</script>

```

CELDOUBLECLICK.CS

```

public ActionResult CellDoubleClick()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}

```

CellRender

The [CellRender](#) event will be triggered before each HeatMap cell is rendered.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").CellRender("heatmapCellRender").TitleSettings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapCellRender(args) {
        console.log("The cell render event has been triggered!!!");
    }

```

```
}
</script>
```

CELLRENDER.CS

```
public ActionResult CellRender()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```

CellSelected

When single or multiple cells in the HeatMap are selected, the [CellSelected](#) event is triggered.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").CellSelected("heatmapCellSelected").AllowSe
lection(true).TitleSettings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
```

```
<script>
function heatmapCellSelected(args) {
    console.log("The cell selected event has been triggered!!!");
}
</script>
```

CELLSELECTED.CS

```
public ActionResult CellSelected()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[, ]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}
```

Created

Once HeatMap has been completely rendered, the [Created](#) event is triggered.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Created("heatmapCreated").TitleSettings(ts
=>
```

```

ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
function heatmapCreated(args) {
    console.log("The created event has been triggered!!!");
}
</script>

```

CREATED.CS

```

public ActionResult Created()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}

```

LegendRender

The [LegendRender](#) event is triggered before the legend is rendered.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").LegendRender("heatmapLegendRender").TitleSettings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapLegendRender(args) {
        console.log("The legend render event has been triggered!!!");
    }
</script>

```

LEGENDRENDER.CS

```

public ActionResult LegendRender()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[, ]
    {
        { 73, 39, 26, 39, 94, 0 },
        { 93, 58, 53, 38, 26, 68 },
        { 99, 28, 22, 4, 66, 90 },
        { 14, 26, 97, 69, 69, 3 },
        { 7, 46, 47, 47, 88, 6 },
        { 41, 55, 73, 23, 3, 79 },
        { 56, 69, 21, 86, 3, 33 },
        { 45, 7, 53, 81, 95, 79 },
        { 60, 77, 74, 68, 88, 51 },
        { 25, 25, 10, 12, 78, 14 },
        { 25, 56, 55, 58, 12, 82 },
        { 74, 33, 88, 23, 86, 59 }
    };
    return data;
}

```

Load

The [Load](#) event is triggered before the HeatMap is rendered.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Load("heatmapLoad").TitleSettings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapLoad(args) {
        console.log("The load event has been triggered!!!");
    }
</script>
```

LOAD.CS

```
public ActionResult Load()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
    return data;
}
```



```
}

```

Loaded

Once HeatMap is loaded, the [Loaded](#) event is triggered.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Loaded("heatmapLoaded").TitleSettings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapLoaded(args) {
        console.log("The loaded event has been triggered!!!");
    }
</script>
```

LOADED.CS

```
public ActionResult Loaded()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]{
        {
            {73, 39, 26, 39, 94, 0},
            {93, 58, 53, 38, 26, 68},
            {99, 28, 22, 4, 66, 90},
            {14, 26, 97, 69, 69, 3},
            {7, 46, 47, 47, 88, 6},
            {41, 55, 73, 23, 3, 79},
            {56, 69, 21, 86, 3, 33},
            {45, 7, 53, 81, 95, 79},
            {60, 77, 74, 68, 88, 51},
            {25, 25, 10, 12, 78, 14},
            {25, 56, 55, 58, 12, 82},
            {74, 33, 88, 23, 86, 59}
        }
    };
}
```

```
};
    return data;
}
```

Resized

When the window is resized, the [Resized](#) event is triggered to notify the resize of the HeatMap.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").Resized("heatmapResized").TitleSettings(ts
=>
    ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
    xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
    yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapResized(args) {
        console.log("The resized event has been triggered!!!");
    }
</script>
```

RESIZED.CS

```
public ActionResult Resized()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
```

```

        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

TooltipRender

The [TooltipRender](#) event is triggered before the tooltip is rendered on the HeatMap cell.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TooltipRender("heatmapTooltipRender").Title
Settings(ts =>
    ts.Text("Sales Revenue per Employee (in 1000
    US$)").TextStyle(ViewBag.textStyle).XAxis(xAxis =>
        xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
            yaxis.Labels(ViewBag.yLabels)).DataSource(ViewBag.dataSource).Render()
<script>
    function heatmapTooltipRender(args) {
        console.log("The tooltip render event has been triggered!!!");
    }
</script>

```

TOOLTIPRENDER.CS

```

public ActionResult TooltipRender()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs", "Fri",
    "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[, ]
    {
        {73, 39, 26, 39, 94, 0},
        {93, 58, 53, 38, 26, 68},
        {99, 28, 22, 4, 66, 90},
        {14, 26, 97, 69, 69, 3},
        {7, 46, 47, 47, 88, 6},
        {41, 55, 73, 23, 3, 79},
    }
}

```

```

        {56, 69, 21, 86, 3, 33},
        {45, 7, 53, 81, 95, 79},
        {60, 77, 74, 68, 88, 51},
        {25, 25, 10, 12, 78, 14},
        {25, 56, 55, 58, 12, 82},
        {74, 33, 88, 23, 86, 59}
    };
    return data;
}

```

How To

Customizing tooltip as a table

You can show a tooltip as a table using the `template` property in `TooltipSettings`.

The following steps describe how to show the table tooltip.

Step 1: Initialize the tooltip template div as shown in the following html page.

Step 2: Set the element id to the `template` property in `TooltipSettings` to show the tooltip template.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts => ts.Text("Crude Oil
Production of Non-OPEC Countries (in Million barrels per
day)").TextStyle(ViewBag.textStyle)).XAxis(
xAxis =>{ xAxis.Labels(ViewBag.xLabels) }).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).CellSettings(cs =>
cs.ShowLabel(false)).TooltipSettings(ts =>

ts.Template("#tooltipTemplate")).ShowTooltip(true).DataSource(ViewBag.dataSo
urce).Render()
<script id="tooltipTemplate" type="text/x-template">
    <div id='template'>
        <table>
            <tr bgcolor="pink"><td
align="right">${xValue}</td><td>${yValue}</td><td >${value}</td></tr>
        </table>
    </div>
</script>

```

TOOLTIP.CS

```

public ActionResult Tooltip()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[8] { "Canada", "China", "Egypt",
    "Mexico", "Norway", "Russia", "UK", "USA"};
    ViewBag.xLabels = xlabels;
}

```

```

        string[] yLabels = new string[11] { "2000", "2001", "2002",
        "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010" };
        ViewBag.yLabels = yLabels;
        ViewBag.dataSource = GetDataSource();
        return View();
    }
    private int[,] GetDataSource()
    {
        int[,] data = new int[,]{
            {0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58,
0.56],
            {2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32,
1.23],
            {2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57,
2.74],
            {3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64,
2.61],
            {3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06,
1.87],
            {3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79,
4.07],
            {5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35,
5.47],
            {6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]
        };
        return data;
    }
}

```

Change the legend label text

You can change the legend label using the **LegendRender** client-side event. You can also hide the legend label using this client-side event.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().HeatMap("container").TitleSettings(ts =>
ts.Text("Sales Revenue per Employee (in 1000
US$)").TextStyle(ViewBag.textStyle)).XAxis(xAxis =>
xAxis.Labels(ViewBag.xLabels)).YAxis(yaxis =>
yaxis.Labels(ViewBag.yLabels)).PaletteSettings(ps => ps.Palette(palette =>
{
    palette.Value(0).Color("#C2E7EC").Add();
    palette.Value(25000).Color("#AEDFE6").Add();
    palette.Value(50000).Color("#9AD7E0").Add();
    palette.Value(75000).Color("#72C7D4").Add();
    palette.Value(99000).Color("#5EBFCE").Add();
}))).LegendSettings(ls =>
ls.Position(Syncfusion.EJ2.HeatMap.LegendPosition.Right)).LegendRender("lege
ndRender")DataSource(ViewBag.dataSource).Render()
<script>
    var legendRender = function (args) {
        if(args.text=='25,000' || args.text=='50,000' ||
args.text=='99,000'){
            args.text = args.text.replace(/,/g, "");
            args.text = `${parseInt(args.text/1000)}` + "k " + "$";
        }
    }

```

```

    } else {
        args.cancel=true;
    }
};
</script>

```

DEFAULT.CS

```

public ActionResult Legend()
{
    ViewBag.textStyle = new
    {
        size= "15px",
        fontWeight= "500",
        fontStyle= "Normal",
        fontFamily= "Segoe UI"
    };
    string[] xlabels = new string[12] { "Nancy", "Andrew", "Janet",
    "Margaret", "Steven", "Michael", "Robert", "Laura", "Anne", "Paul", "Karin",
    "Mario" };
    ViewBag.xLabels = xlabels;
    string[] yLabels = new string[6] { "Mon", "Tues", "Wed", "Thurs",
    "Fri", "Sat" };
    ViewBag.yLabels = yLabels;
    ViewBag.dataSource = GetDataSource();
    return View();
}
private int[,] GetDataSource()
{
    int[,] data = new int[,]
    {
        {73000, 39000, 26000, 39000, 94000, 0},
        {93000, 58000, 53000, 38000, 26000, 68000},
        {99000, 28000, 22000, 4000, 66000, 9000},
        {14000, 26000, 97000, 69000, 69000, 3000},
        {7000, 46000, 47000, 47000, 88000, 6000},
        {41000, 55000, 73000, 23000, 3000, 79000},
        {56000, 69000, 21000, 86000, 3000, 33000},
        {45000, 7000, 53000, 81000, 95000, 79000},
        {60000, 77000, 74000, 68000, 88000, 51000},
        {25000, 25000, 10000, 12000, 78000, 14000},
        {25000, 56000, 55000, 58000, 12000, 82000},
        {74000, 33000, 88000, 23000, 86000, 59000}
    };
    return data;
}

```

Migration from Essential JS 1

This article describes the API migration process of heat map component from Essential JS 1 to Essential JS 2.

Members

| Behaviour | API in Essential JS 1 | API in Essential JS 2 |

| ---- | --- | --- |

| Specifies the width of the heat map | **Property:** `width`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 Heatmap.Width = "830"; | **Property:**
`width`
 @Html.EJS().HeatMap("container").Width("300px").Render()

| Specifies the height of the heat map | **Property:** `height`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 Heatmap.Height = "830"; | **Property:** `height`
 @Html.EJS().HeatMap("container").Height("300px").Render() |

| Enables or disables tooltip of heat map | **Property:**
`showtooltip`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 Heatmap.ShowTooltip = true; | **Property:**
`showtooltip`
 @Html.EJS().HeatMap("container").ShowTooltip(true).Render() |

| Defines the tooltip that should be shown when the mouse hovers over cells. | **Property:**
`toolTipSettings.templateId`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 Heatmap.ToolTipSettings = new ToolTipSettings() {
 templateId = "mouseovertoolTipId" }; | **Property:**
`tooltipRender`
 @Html.EJS().HeatMap("container").TooltipRender("tooltipRender").Render()
 >var tooltipRender = function (args) {};

| Specifies the source data of the heat map. | **Property:**
`itemsSource`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 Heatmap.ItemsSource = []; | **Property:**
`dataSource`
 @Html.EJS().HeatMap("container").DataSource(ViewBag.dataSource).Render() |

| Specifies whether the cell content can be visible or not. | **Property:**
`heatmapCell.showContent`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 Heatmap.HeatMapCell = new HeatMapCell() { showContent =
 "Hidden" }; | **Property:**
`cellSettings.showLabel`
 @Html.EJS().HeatMap("container").CellSettings(cs =>
 cs.ShowLabel(false)).Render() |

| Specifies the color of the heat map column data. | **Property:**
`colorMappingCollection.color`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
 Heatmap = new HeatMapProperties();
 List colorCollection = new List();
 colorCollection.Add(new HeatMapColorMapping() { Color = "#8ec8f8" });
 Heatmap.ColorMappingCollection = colorCollection;
 | **Property:**
`paletteSettings.palette.color`
 @Html.EJS().HeatMap("container").PaletteSettings(ps =>
 ps.Palette(palette => { palette.Value(0).Color("rgb(238,238,238)").Add(); })).Render() |

| Specifies the color values of the heat map column data. | **Property:** colorMappingCollection.value

@Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"] as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)

HeatMapProperties
 Heatmap = new HeatMapProperties(); List colorCollection = new List();
colorCollection.Add(new
 HeatMapColorMapping() { Value = 0 });
Heatmap.ColorMappingCollection = colorCollection; |

Property:

paletteSettings.palette.value

@Html.EJS().HeatMap("container").PaletteSettings(ps =>
 ps.Palette(palette => { palette.Value(20).Add(); })).Render() |

| Specifies the label text of the heat map color. | **Property:**

colorMappingCollection.label.text

@Html.EJ().HeatMap("heatmap",
 ViewData["HeatMapModel"] as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)

HeatMapProperties
 Heatmap = new HeatMapProperties(); List colorCollection = new List();
colorCollection.Add(new
 HeatMapColorMapping() { Label = new HeatMapLabel() { Text = "Moderate" }
 });
Heatmap.ColorMappingCollection = colorCollection; | **Property:**
 paletteSettings.palette.label

@Html.EJS().HeatMap("container").PaletteSettings(ps =>
 ps.Palette(palette => { palette.Label("no contributions").Add(); })).Render() |

| Specifies the style of the heat map color label. | **Property:** colorMappingCollection.label.bold

Property: colorMappingCollection.label.italic

@Html.EJ().HeatMap("heatmap",
 ViewData["HeatMapModel"] as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)

HeatMapProperties
 Heatmap = new HeatMapProperties();
List colorCollection = new List();
colorCollection.Add(new
 HeatMapColorMapping() { Label = new HeatMapLabel() { Bold = true }
 });
Heatmap.ColorMappingCollection = colorCollection; | **Property:**
 legendSettings.textStyle.fontStyle

@Html.EJS().HeatMap("container").LegendSettings(ls =>
 ls.TextStyle(ViewBag.textStyle)).Render()

ViewBag.textStyle new { fontStyle:'bold' }; |

| Specifies the font size of the heat map label. | **Property:**

colorMappingCollection.label.fontSize

@Html.EJ().HeatMap("heatmap",
 ViewData["HeatMapModel"] as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)

HeatMapProperties
 Heatmap = new HeatMapProperties(); List colorCollection = new List();
colorCollection.Add(new
 HeatMapColorMapping() { Label = new HeatMapLabel() { FontSize = 18 }
 });
Heatmap.ColorMappingCollection = colorCollection; | **Property:**
 legendSettings.textStyle.size

@Html.EJS().HeatMap("container").LegendSettings(ls =>
 ls.TextStyle(ViewBag.textStyle)).Render()

ViewBag.textStyle = new { size: 18 }; |

| Specifies the font family of the heat map label. | **Property:**

colorMappingCollection.label.fontFamily

@Html.EJ().HeatMap("heatmap",
 ViewData["HeatMapModel"] as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)

HeatMapProperties
 Heatmap = new HeatMapProperties(); List colorCollection = new List();
colorCollection.Add(new
 HeatMapColorMapping() { Label = new HeatMapLabel() { FontFamily = "Arial" }
 });
Heatmap.ColorMappingCollection = colorCollection; | **Property:**
 legendSettings.textStyle.fontFamily

@Html.EJS().HeatMap("container").LegendSettings(ls =>
 ls.TextStyle(ViewBag.textStyle)).Render()

ViewBag.textStyle = new { fontFamily: 'Arial' } |

| Specifies the font color of the heat map label. | **Property:**
`colorMappingCollection.label.fontColor`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties
 Heatmap = new HeatMapProperties();
 List colorCollection = new List();
 colorCollection.Add(new HeatMapColorMapping() { Label = new HeatMapLabel() { FontColor = "red" } });
 Heatmap.ColorMappingCollection = colorCollection; | **Property:**
`legendSettings.textStyle.fontFamily`
 @Html.EJS().HeatMap("container").LegendSettings(ls => ls.TextStyle(ViewBag.textStyle)).Render()
 ViewBag.textStyle = new { color: 'red' } |

| Specifies the mapping name of the column. | **Property:**
`itemsMapping.column.propertyName`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties
 Heatmap = new HeatMapProperties();
 TableMapping TableMapping = new TableMapping();
 TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2010" });
 Heatmap.ItemsMapping = TableMapping; | **Property:**
`dataSource.yDataMapping`
 @Html.EJS().HeatMap("container").DataSource(ViewBag.dataSource).Render()
 ViewBag.dataSource = new { data: heatmapData, yDataMapping: 'columnid' }; |

| Specifies the mapping name of the row. | **Property:**
`itemsMapping.row.propertyName`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties
 Heatmap = new HeatMapProperties();
 TableMapping TableMapping = new TableMapping();
 TableMapping.RowMapping.Add(new HeaderMapping() { PropertyName = "Y2010" });
 Heatmap.ItemsMapping = TableMapping; | **Property:**
`dataSource.xDataMapping`
 @Html.EJS().HeatMap("container").DataSource(ViewBag.dataSource).Render()
 ViewBag.dataSource = new { data: heatmapData, xDataMapping: 'rowid' }; |

| Specifies the mapping name of the row. | **Property:**
`itemsMapping.value.propertyName`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties
 Heatmap = new HeatMapProperties();
 TableMapping TableMapping = new TableMapping();
 TableMapping.ValueMapping.Add(new HeaderMapping() { PropertyName = "Y2010" });
 Heatmap.ItemsMapping = TableMapping; | **Property:**
`dataSource.valueMapping`
 @Html.EJS().HeatMap("container").DataSource(ViewBag.dataSource).Render()
 ViewBag.dataSource = new { data: heatmapData, valueMapping: 'value' }; |

Events

| Behaviour | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggered when the cell get clicked. | **Property:**
`cellSelected`
 @Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"]) as
 Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties
 Heatmap = new HeatMapProperties();
 Heatmap.CellSelected = function(args) {}; | **Property:**

```
cellClick<br><br>@Html.EJS().HeatMap("container").CellClick("cellClick").Render()<br><br>var cellClick  
= function (args) {};
```

In-place Editor

Getting Started with ASP.NET MVC In-place Editor Control

This section briefly explains about how to include [ASP.NET MVC In-place Editor](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>  
<add namespace="Syncfusion.EJ2"/>  
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>  
...
```

```
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC In-place Editor control

Now, add the Syncfusion ASP.NET MVC In-place Editor control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().InPlaceEditor("element").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC In-place Editor control will be rendered in the default web browser.

Add the In-place Editor with Textbox

By default, the Syncfusion ASP.NET MVC TextBox control is rendered in In-place Editor with the [Type](#) property sets as Text.

CSHTML

```
<div style="margin:0 auto; width:300px;">

@Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
Type.Text).Value(ViewBag.value).Model(ViewBag.modalData).Render()
</div>
```

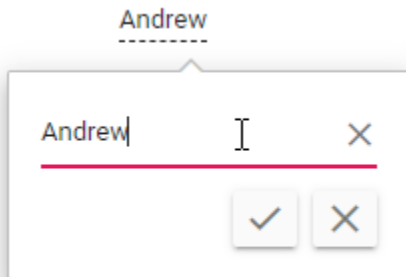
HOMECONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
```

```

    ViewBag.modalData = new { placeholder = "Enter employee name" };
    ViewBag.value = "Andrew";
    return View();
}

```



Configuring DropDownList

You can render the Syncfusion ASP.NET MVC DropDownList by changing the [Type](#) property as [DropDownList](#) and configure its properties and methods using the [Model](#) property.

In the following sample, [Type](#) and [Model](#) values are configured to render the [DropDownList](#) control.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor
<div style="margin:0 auto; width:300px;">

@Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
Type.DropDownList).Mode(RenderMode.Inline).Model(ViewBag.modalData).Render()
</div>

```

HOMECONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        string[] genderData = new string[] { "Male", "Female" };
        ViewBag.modalData = new { placeholder = "Select gender", dataSource
= genderData };
        return View();
    }
}

```



Integrate DatePicker

You can render the Essential JS2 DatePicker by changing the [Type](#) property as `Date` and also configure its properties and methods using the [Model](#) property.

In the following sample, `Type` and `Model` values are configured to render the `DatePicker` control.

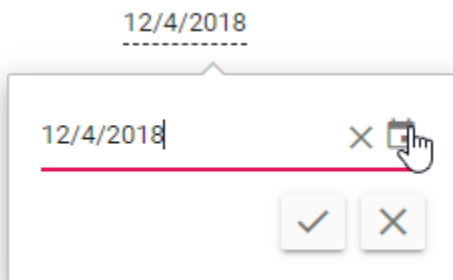
CSHTML

```
<div style="margin:0 auto; width:300px;">

@Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
Type.Date).Value(ViewBag.dateValue).Model(ViewBag.modalData).Render()
</div>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.dateValue = new DateTime(2018, 12, 04);
        ViewBag.modalData = new { showTodayButton = true };
        return View();
    }
}
```



In the following sample, type and model values are configured to render the `TextBox`, `DropDownList` and `DatePicker` control.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor
<div id='container'>
    <div class="control-group">
        <div class="e-heading">
            <h3> Modify Basic Details </h3>
        </div>
        <table>
            <tr>
                <td>Name</td>
                <td class='left'>
```

```
@Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
```

```

Type.Text).Value(ViewBag.elementValue).Model(ViewBag.elementModel).Mode(RenderMode.Inline).Render()
        </td>
    </tr>
    <tr>
        <td>Date of Birth</td>
        <td class='left'>

@Html.EJS().InPlaceEditor("dateofbirth").Type(Syncfusion.EJ2.InPlaceEditor.InputType.Date).Value(ViewBag.dateValue).Model(ViewBag.dateModel).Mode(RenderMode.Inline).Render()
        </td>
    </tr>
    <tr>
        <td>Gender</td>
        <td class='left'>

@Html.EJS().InPlaceEditor("gender").Type(Syncfusion.EJ2.InPlaceEditor.InputType.DropDownList).Value(ViewBag.dropValue).Model(ViewBag.genderModel).Mode(RenderMode.Inline).Render()
        </td>
    </tr>
</table>
</div>
</div>
<style>
    #container .control-group {
        margin-top: 50px;
    }
    #container .control-group table {
        margin: auto;
    }
    #container .control-group table td {
        height: 70px;
        width: 150px;
    }
    #container .e-heading {
        text-align: center;
    }
    #container .control-group table td {
        text-align: left;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.elementModel = new { placeholder = "Enter your name" };
        ViewBag.dateValue = new DateTime(2018, 12, 04);
        ViewBag.dateModel = new { showTodayButton = true, placeholder = "Select date of birth" };
        string[] genderData = new string[] { "Male", "Female" };
    }
}

```

```

        ViewBag.genderModel = new { placeholder = "Select gender",
        dataSource = genderData };
        ViewBag.elementValue = "Andrew";
        ViewBag.dropValue = "Male";
        return View();
    }
}

```

Modify Basic Details

Name	Andrew -----
Date of Birth	12/4/2018 -----
Gender	Male -----

Submitting data to the server (save)

You can submit editor value to the server by configuring the [Url](#), [Adaptor](#) and [PrimaryKey](#).

Property	Usage
Url	Gets the URL for server submit action.
Adaptor	Specifies the adaptor type that is used by DataManager to communicate with DataSource.
PrimaryKey	Defines the unique primary key of editable field which can be used for saving data in the data-base.

Note: The **PrimaryKey** property is mandatory. If it's not set, edited data are not sent to the server.

Refresh with modified value

The edited data is submitted to the server and you can see the new values getting reflected in the In-place Editor.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor
<div id='container'>
    <div className='control-group' style="text-align: center; width: 50%;
margin: 100px auto;">
        Best Employee of the year:

    @Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
Type.DropDownList).Value(ViewBag.textValue).Model(ViewBag.elementModel).Mode

```

```

(RenderMode.Inline).Name("Employee").Url(ViewBag.url).PrimaryKey("Employee")
.Adaptor(AdaptorType.UrlAdaptor).ActionSuccess("actionSuccess").Render()
</div>
<table style='margin:auto;width:50%'>
  <tr>
    <td style="text-align: left">
      Old Value :
    </td>
    <td id="oldValue" style="text-align: left"></td>
  </tr>
  <tr>
    <td style="text-align: left">
      New Value :
    </td>
    <td id="newValue" style="text-align: left">
      Andrew Fuller
    </td>
  </tr>
</table>
</div>
<style>
.e-inplaceeditor {
  min-width: 200px;
  text-align: left
}
#container .control-group {
  text-align: center;
  margin: 100px auto;
}
</style>
<script>
function actionSuccess(e) {
  var newEle = document.getElementById('newValue');
  var oldEle = document.getElementById('oldValue');
  oldEle.textContent = newEle.textContent;
  newEle.textContent = e.value;
}
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
  public ActionResult Index()
  {
    string[] frameWorkList = new string[] { "Andrew Fuller", "Janet
Leverling", "Laura Callahan", "Margaret Hamilt", "Michael Suyama", "Nancy
Davloio", "Robert King" };
    ViewBag.elementModel = new { dataSource = frameWorkList, placeholder
= "Select employee", popupHeight = "200px" };
    ViewBag.url = "https://ej2services.syncfusion.com/development/web-
services/api/Editor/UpdateData";
    ViewBag.textValue = "Andrew Fuller";
    return View();
  }
}

```


Best Employee of the year: Margaret Hamilt

Old Value : Andrew Fuller
New Value : Margaret Hamilt

Note: [View Sample in GitHub.](#)

See also

- [Real time example using In-place Editor](#)
- [Types of rendering the editor](#)

List of controls

In-place Editor renders various controls based on the [Type](#) property and it have built-in and injectable controls. To use injectable controls, inject the required modules into `InPlaceEditor`. By default, the `type` property set to `Text` and render the `TextBox`.

The following table explains Injectable components module name and built-in components and their types.

Injectable Components Built in Components	
----- -----	
AutoComplete (AutoComplete)	TextBox (Text)
ComboBox (ComboBox)	DatePicker (Date)
MultiSelect (MultiSelect)	DateTimePicker (DateTime)
TimePicker (Time)	DropDownList (DropDownList)
DateRangePicker (DateRange)	MaskedTextBox (Mask)
Slider (Slider)	NumericTextBox (Numeric)
Rte (RTE)	
ColorPicker (Color)	

In the following sample, built-in and injectable based In-place Editor controls are rendered.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor
<div id='container'>
  <h3> Built-in Controls </h3>
  <table class="table-section">
```

```

        <tr>
            <td class="col-lg-6 control-title"> DatePicker </td>
            <td class="col-lg-6">

@Html.EJS().InPlaceEditor("date").Mode(RenderMode.Inline).Type(Syncfusion.EJ
2.InPlaceEditor.InputType.Date).Value(ViewBag.dateValue).Model(ViewBag.dateM
odelData).Render()

            </td>
        </tr>
        <tr>
            <td class="col-lg-6 control-title"> DateTimePicker </td>
            <td class="col-lg-6">

@Html.EJS().InPlaceEditor("dateTime").Mode(RenderMode.Inline).Type(Syncfusio
n.EJ2.InPlaceEditor.InputType.DateTime).Value(ViewBag.dateTimeValue).Model(V
iewBag.dateModelData).Render()

            </td>
        </tr>
        <tr>
            <td class="col-lg-6 control-title"> DropDownList </td>
            <td class="col-lg-6">

@Html.EJS().InPlaceEditor("dropDowns").Mode(RenderMode.Inline).Type(Syncfusi
on.EJ2.InPlaceEditor.InputType.DropDownList).Value(ViewBag.dropDownValue).Mo
del(ViewBag.dropModelData).Render()

            </td>
        </tr>
        <tr>
            <td class="col-lg-6 control-title"> MaskedTextBox </td>
            <td class="col-lg-6">

@Html.EJS().InPlaceEditor("masked").Mode(RenderMode.Inline).Type(Syncfusion.
EJ2.InPlaceEditor.InputType.Mask).Value(ViewBag.maskValue).Model(ViewBag.mas
kModelData).Render()

            </td>
        </tr>
        <tr>
            <td class="col-lg-6 control-title"> NumericTextBox </td>
            <td class="col-lg-6">

@Html.EJS().InPlaceEditor("numeric").Mode(RenderMode.Inline).Type(Syncfusion
.EJ2.InPlaceEditor.InputType.Numeric).Value(10).Model(ViewBag.numericModelDa
ta).Render()

            </td>
        </tr>
        <tr>
            <td class="col-lg-6 control-title"> TextBox </td>
            <td class="col-lg-6">

@Html.EJS().InPlaceEditor("textbox").Mode(RenderMode.Inline).Type(Syncfusion
.EJ2.InPlaceEditor.InputType.Text).Value(ViewBag.textValue).Model(ViewBag.te
xtModelData).Render()

            </td>
        </tr>
    </table>
    <h3> Injectable Controls </h3>
    <table class="table-section">

```

```

<tr>
  <td class="col-lg-6 control-title"> AutoComplete </td>
  <td class="col-lg-6">

@Html.EJS().InPlaceEditor("autoComplete").Mode(RenderMode.Inline).Type(Syncf
usion.EJ2.InPlaceEditor.InputType.AutoComplete).Value(ViewBag.dropDownValue)
.Model(ViewBag.dropModelData).Render()

  </td>
</tr>
<tr>
  <td class="col-lg-6 control-title"> ColorPicker </td>
  <td class="col-lg-6">

@Html.EJS().InPlaceEditor("color").Mode(RenderMode.Inline).Type(Syncfusion.E
J2.InPlaceEditor.InputType.Color).Value(ViewBag.colorValue).Render()

  </td>
</tr>
<tr>
  <td class="col-lg-6 control-title"> ComboBox </td>
  <td class="col-lg-6">

@Html.EJS().InPlaceEditor("comboBox").Mode(RenderMode.Inline).Type(Syncfusio
n.EJ2.InPlaceEditor.InputType.ComboBox).Value(ViewBag.dropDownValue).Model(V
iewBag.dropModelData).Render()

  </td>
</tr>
<tr>
  <td class="col-lg-6 control-title"> DateRangePicker </td>
  <td class="col-lg-6">

@Html.EJS().InPlaceEditor("dateRange").Mode(RenderMode.Inline).Type(Syncfusi
on.EJ2.InPlaceEditor.InputType.DateRange).Value(ViewBag.dateRangeValue).Mode
l(ViewBag.dateModelData).Render()

  </td>
</tr>
<tr>
  <td class="col-lg-6 control-title"> MultiSelect </td>
  <td class="col-lg-6">

@Html.EJS().InPlaceEditor("multiSelect").Mode(RenderMode.Inline).Type(Syncfu
sion.EJ2.InPlaceEditor.InputType.MultiSelect).Value(ViewBag.dropDownValue).M
odel(ViewBag.dropModelData).Render()

  </td>
</tr>
<tr>
  <td class="col-lg-6 control-title"> RTE </td>
  <td class="col-lg-6">

@Html.EJS().InPlaceEditor("rte").Mode(RenderMode.Inline).Type(Syncfusion.EJ2
.InPlaceEditor.InputType.RTE).Value("Enter your content
here").Model(ViewBag.rteModelData).Render()

  </td>
</tr>
<tr>
  <td class="col-lg-6 control-title"> Slider </td>
  <td class="col-lg-6">
    <div id="slider"></div>

```

```

@Html.EJS().InPlaceEditor("slider").Mode(RenderMode.Inline).Type(Syncfusion.
EJ2.InPlaceEditor.InputType.Slider).Value(20).Render()
    </td>
</tr>
<tr>
    <td class="col-lg-6 control-title"> TimePicker </td>
    <td class="col-lg-6">

@Html.EJS().InPlaceEditor("time").Mode(RenderMode.Inline).Type(Syncfusion.EJ
2.InPlaceEditor.InputType.Time).Value(ViewBag.dateValue).Model(ViewBag.dateM
odelData).Render()
    </td>
</tr>
</table>
</div>
<style>
    body {
        padding: 20px 0
    }
    .control-title {
        font-weight: 600;
        padding-right: 20px;
    }
    td {
        height: 80px;
    }
    tr td:first-child {
        text-align: right;
    }
    tr td:last-child {
        text-align: left;
    }
    .table-section {
        margin: 0 auto;
    }
    h3 {
        text-align: center;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        var frameWorkList = new string[] { "Android", "JavaScript",
"jQuery", "TypeScript", "Angular", "React", "Vue", "Ionic" };
        ViewBag.dateValue = new DateTime(2018, 11, 23);
        ViewBag.dateTimeValue = new DateTime(2018, 11, 23, 12, 30, 00);
        ViewBag.dateRangeValue = new DateTime[] { new DateTime(2018, 11,
12), new DateTime(2018, 11, 15) };
        ViewBag.dateModelData = new { placeholder = "Select date" };
        ViewBag.dropModelData = new { dataSource = frameWorkList,
placeholder = "Select frameworks" };
    }
}

```

```

ViewBag.maskModelData = new { mask = "000-000-000" };
ViewBag.rteModelData = new { placeholder = "Enter your content here"
};

ViewBag.numericModelData = new { placeholder = "Enter number" };
ViewBag.textModelData = new { placeholder = "Enter some text" };
ViewBag.dropDownValue = "Android";
ViewBag.maskValue = "123-345-678";
ViewBag.textValue = "Andrew";
ViewBag.colorValue = "#81aefd";
ViewBag.rteValue = "<p>Enter your content here</p>";
return View();
}
}

```

The output will be as follows.

Built-in Controls

DatePicker 11/22/2018

DateTimePicker 11/21/2018 1:30 PM

DropDownList logic

MaskedTextBox 242-423-423

NumericTextBox 2.00

TextBox Andrew ffsafsf

Injectable Controls

AutoComplete React

ColorPicker #244a8d

ComboBox Ionic

DateRangePicker 12/12/2018 - 12/20/2018

MultiSelect Android,React,Ionic,TypeScript

RTE afasfsfsf

Slider 34

TimePicker 2:30 AM

Model configuration

Control properties and events can be customized using the In-place Editor [Model](#) property.

In the following code, the [Type](#) defined as the `Date` and `DatePicker` properties are configured through `Model` property to customize the `DatePicker` control at In-place Editor.

CSHTML

```
<div style="margin:0 auto; width:300px;">

@Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
Type.Date).Value(ViewBag.dateValue).Model(ViewBag.dateModelData).Render()
</div>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.dateValue = new DateTime(2018, 12, 04);
        ViewBag.dateModelData = new { showTodayButton = true, placeholder =
"Select date" };
        return View();
    }
}
```

See Also

- [Disable the editor](#)
- [Animate the editor during popup mode](#)

Configuration

Rendering modes

This section explains the supported rendering modes of the In-place Editor. Possible Rendering modes are as follows.

- Popup
- Inline

Note: By default, `Popup` mode will be rendered, when opening an editor.

- For `Popup` mode, editable container displays as like tooltip or popover above the element.
- For `Inline` mode, editable container displays as instead of the element. To render `Inline` mode while opening the editor, specify [Mode](#) as `Inline`.

In the following sample, the In-place Editor renders with `Inline` mode. You can dynamically switch into another mode by changing the drop-down item value.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> Mode: </td>
            <td>

@Html.EJS().DropDownList("dropDown").Width("auto").DataSource(ViewBag.modeData).Value(ViewBag.value).Change("onChange").Placeholder("Select Mode").Render()
            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value("Andrew").Model(ViewBag.modelData).Render()
            </td>
        </tr>
    </table>
</div>
<style>
    .table-section {
        margin: 0 auto;
    }
    tr td:first-child {
        text-align: right;
        padding-right: 20px;
    }
    .sample-td {
        padding-top: 10px;
        min-width: 230px;
        height: 100px;
    }
</style>
<script>
    function onChange(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        var mode = e.itemData.value;
        editObj.mode = mode;
        editObj.dataBind();
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.modeData = new string[] { "Inline", "Popup" };
        ViewBag.value = "Inline";
        ViewBag.modalData = new { placeholder = "Enter some text" };
        return View();
    }
}

```

```
}

```

The output will be as follows.

Mode: Inline ▼

Enter your name: Andrew Fuller ✕

✓ ✕

Pop-up customization

In-place Editor popup mode can be customized by using the **Title** and **Model** properties in [PopupSettings](#) API.

Popup mode rendered by using the Syncfusion ASP.NET MVC Tooltip control, so you can use tooltip properties and events to customize the behavior of popup via the **Model** property of **PopupSettings** API.

Note: For more details, refer the tooltip documentation section.

In the following sample, popup **Title** and **Position** customized using the **PopupSettings** property. All possible tooltip position data configured in the drop-down, if we change drop down item, selected value bound to **Model** property and applied it to **Tooltip** control. **Tooltip** have following position options.

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop
- RightCenter
- RightBottom

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> Position: </td>
            <td>
                @Html.EJS().DropDownList("dropDown").DataSource(ViewBag.positionData).Value(
```



```

ViewBag.text).Change("onChange").Placeholder("Select a
position").PopupHeight("150px").Render()
    </td>
</tr>
<tr>
    <td class="edit-heading sample-td"> Enter your name: </td>
    <td class="sample-td">

@Html.EJS().InPlaceEditor("element").PopupSettings(e=>e.Title("Enter
name").Model(ViewBag.model)).Mode(RenderMode.Popup).Value(ViewBag.textValue)
.Model(ViewBag.modelData).Render()
    </td>
</tr>
</table>
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
    }
    .table-section {
        margin: 0 auto;
    }
    .sample-td {
        padding-top: 150px;
    }
    .edit-heading {
        padding-right: 20px;
    }
</style>
<script>
    function onChange(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.popupSettings.model.position = e.value;
        editObj.dataBind();
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.positionData = new string[] { "TopLeft", "TopCenter",
"TopRight", "BottomLeft", "BottomCenter", "BottomRight", "LeftTop",
"LeftCenter", "LeftBottom", "RightTop", "RightCenter", "RightBottom" };
        ViewBag.modelData = new { placeholder = "Enter some text" };
        ViewBag.text = "BottomCenter";
        ViewBag.textValue = "Andrew";
        ViewBag.model = new { position = "BottomCenter" };
        return View();
    }
}

```

The output will be as follows.

Position: TopCenter ▼

Enter name

Andrew

✓ ✕

Enter your name: Andrew

Event actions for editing

The event action of the editor that enable in the edit mode based on the [EditableOn](#) property, by default **Click** is assigned, the following options are also supported.

- **Click**: The editor will be opened as single click actions.
- **DbClick**: The editor will be opened as double-click actions and it is not applicable for edit icon.
- **EditIconClick**: Disables the editing of event action of input and allows user to edit only through edit icon.

Note: In-place Editor get focus by pressing the **tab** key from previous focusable DOM element and then by pressing **enter** key, the editor will be opened.

In the following sample, when switching drop-down item, the selected value assigned to the **EditableOn** property. If you changed to **DbClick**, the editor will open when making a double click on the input.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> EditableOn: </td>
            <td>
                @Html.EJS().DropDownList("dropDown").Width("auto").DataSource(ViewBag.editableOnData).Value(ViewBag.text).Change("onChange").Placeholder("Select edit type").Render()
            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">
                @Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.textValue).Locale("fr-BE").Model(ViewBag.modelData).Render()
            </td>
        </tr>
    </table>
</div>
```

```

        </td>
    </tr>
</table>
</div>
<style>
    .table-section {
        margin: 0 auto;
    }
    tr td:first-child {
        text-align: right;
        padding-right: 20px;
    }
    .sample-td {
        padding-top: 10px;
        min-width: 230px;
        height: 100px;
    }
</style>
<script>
    function onChange(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        var editType = e.itemData.value;
        editObj.editableOn = editType;
        editObj.dataBind();
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.modelData = new { placeholder = "Enter some text" };
        ViewBag.editableOnData = new string[] { "Click", "DbClick",
"EditIconClick" };
        ViewBag.text = "Click";
        ViewBag.textValue = "Andrew";
        return View();
    }
}

```

The output will be as follows.

EditableOn: Click ▼

Enter your name: Andrew

Action on focus out

Action to be performed when the user clicks outside the container, that means focusing out of editable content and it can be handled by the [ActionOnBlur](#) property, by default **Submit** assigned. It also has the following options.

- **Cancel:** Cancels the editing and resets the old content.
- **Submit:** Submits the edited content to the server.
- **Ignore:** No action is performed with this type and allows to edit multiple editors.

In the following sample, when switching drop-down item, the selected value assigned to the **ActionOnBlur** property.

CSHTML

```
<div id='container'>
  <table class="table-section">
    <tr>
      <td> ActionOnBlur: </td>
      <td>

@Html.EJS().DropDownList("dropDown").DataSource(ViewBag.blurActionData).Width("auto").Value(ViewBag.text).Change("onChange").Placeholder("Select blur action").Render()
      </td>
    </tr>
    <tr>
      <td class="sample-td"> Enter your name: </td>
      <td class="sample-td">

@Html.EJS().InPlaceEditor("element").Mode(Syncfusion.EJ2.InPlaceEditor.RenderMode.Inline).Value(ViewBag.textValue).Model(ViewBag.modelData).Render()
      </td>
    </tr>
  </table>
</div>
<style>
  .table-section {
    margin: 0 auto;
  }
  tr td:first-child {
    text-align: right;
    padding-right: 20px;
  }
  .sample-td {
    padding-top: 10px;
    min-width: 230px;
    height: 100px;
  }
</style>
<script>
  function onChange(e) {
    var editObj = document.getElementById('element').ej2_instances[0];
    let editType = e.itemData.value;
    editObj.actionOnBlur = editType;
    editObj.dataBind();
  }
</script>
```

```
}
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.blurActionData = new string[] { "Submit", "Cancel", "Ignore"
    };

        ViewBag.modelData = new { placeholder = "Enter some text" };
        ViewBag.text = "Submit";
        ViewBag.textValue = "Andrew";
        return View();
    }
}
```

The output will be as follows.

ActionOnBlur: ▼

Enter your name:

Display modes

By default, In-place Editor input element highlighted with a dotted underline. To remove dotted underline from input element, add `data-underline="false"` attribute at In-place Editor root element.

In the following sample, denotes to indicate intractable and normal display modes with different samples.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <h4>Example of data-underline attribute</h4>
    <table class="table-section">
        <tr>
            <td class="col-lg-6 control-title"> Intractable UI </td>
            <td class="col-lg-6">
                @Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.textValue).Model(ViewBag.modelData).Render()
            </td>
        </tr>
        <tr>
            <td class="col-lg-6 control-title"> Normal UI </td>
            <td class="col-lg-6">
```

```
@Html.EJS().InPlaceEditor("inline").Mode(RenderMode.Inline).Value(ViewBag.textValue).Model(ViewBag.modelData).Created("created").Render()

    </td>
  </tr>
</table>
</div>
<style>
  .table-section {
    margin: 0 auto;
  }
  td {
    padding: 20px 0;
    min-width: 230px;
    height: 100px;
  }
  .control-title {
    font-weight: 600;
    padding-right: 20px;
    text-align: right;
  }
  h4 {
    text-align: center;
  }
</style>
<script>
  function created() {
    document.getElementById("inline").setAttribute("data-underline",
    "false");
  }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
  public ActionResult Index()
  {
    ViewBag.modelData = new { placeholder = "Enter some text" };
    ViewBag.textValue = "Andrew";
    return View();
  }
}
```

The output will be as follows.

Example of data-underline attribute

Intractable UI

Andrew

Normal UI

Andrew

Buttons

The In-place Editor had an action for save and cancel using buttons. The [SaveButton](#) and [CancelButton](#) properties accept the `ButtonModel` objects for customizing the save and cancel button properties.

Buttons can be show or hide by sets a Boolean value to the [ShowButtons](#) property.

Note: Without buttons value will be processed via the following ways.

- **ActionOnBlur:** By clicking out side the editor control get focus out and do action based on this property value.
- **SubmitOnEnter:** Pressing `Enter` key it performs the submit action, if this property set to `true`.

In the following sample, the `content` and `cssClass` properties of `Button` value assigned to the `SaveButton` and `CancelButton` properties to customize its appearance. Also check or uncheck a checkbox buttons render or removed from the editor.

To restrict either save or cancel button rendering into a DOM, simply pass empty object `{}` in the `SaveButton` or `CancelButton` properties.

Note: For more details about buttons, refer this documentation section.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> ShowButtons: </td>
            <td>
                @Html.EJS().CheckBox("enableBtn").Change("onChange").Checked(true).Label("Show")
            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">
                @Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.t
```

```

extValue).Model(ViewBag.modelData).SaveButton(ViewBag.saveButton).CancelButt
on(ViewBag.cancelButton).Render()
        </td>
    </tr>
</table>
</div>
<style>

    .table-section {
        margin: 0 auto;
    }
    tr td:first-child {
        text-align: right;
        padding-right: 20px;
    }
    .sample-td {
        padding-top: 10px;
        min-width: 230px;
        height: 100px;
    }
</style>
<script>
    function onChange(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.showButtons = e.checked;
        editObj.dataBind();
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.modalData = new { placeholder = "Enter some text" };
        ViewBag.saveButton = new { content = "Ok", cssClass = "e-outline" };
        ViewBag.cancelButton = new { content = "Cancel", cssClass = "e-
outline" };
        ViewBag.textValue = "Andrew";
        return View();
    }
}

```

The output will be as follows.

ShowButtons: ☒ Show

Enter your name: X

See Also

- [In-place editor buttons](#)

Server actions

By passing In-place Editor control value to the server, the [PrimaryKey](#) property value must require, otherwise action not performed for remote data.

If the [Url](#) property value is empty, data passing will handled at local and also the [ActionSuccess](#) event will trigger with `null` as argument value.

Note: The following arguments are passed to the server when submit actions perform.

Arguments	Explanations
value	For processing edited value, like DB value updating.
primaryKey	For value mapping to the server, like selecting DB.
name	For field mapping to the server, like DB column field mapping.

Find the following sample server codes for defining models and controller functions to configure processing data.

```
`csharp
public class SubmitModel
{
    public string Name { get; set; }
    public string PrimaryKey { get; set; }
    public string Value { get; set; }
}
`

`csharp
public void UpdateData([FromBody]SubmitModel model)
{
    // User can process data here
}
```

- Server actions successfully done, the [ActionSuccess](#) event will be fired with returned server data.
- If the server is not responding, the [ActionFailure](#) event will be fired with data, but value not updated in the Editor.

In the following sample, the `ActionSuccess` event will trigger once the value submitted successfully into the server. In this sample, both `ActionSuccess` and `ActionFailure` were configured and resulted value will be converted to chips.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor
<div id='container'>
    <span class="content-title"> Select frameworks: </span>

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Type(Syncfusion
.EJ2.InPlaceEditor.InputType.MultiSelect).Value(ViewBag.value).Name("skill")
.Url(Url.Action("UpdateData",
"Home")).PrimaryKey("FrameWork").Adaptor(AdaptorType.WebApiAdaptor).Model(Vi
ewBag.model).ActionSuccess("actionSuccess").ActionFailure("actionFailure").C
reated("created").Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80px;
    }
    #element {
        width: 150px;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>
<script>
    function actionSuccess(e) {
        e.value = chipCreation(e.value.split(','), true);
    }
    function actionFailure(e) {
        e.value = chipCreation(e.value.split(','), false);
    }
    function created() {
        chipOnCreate();
        document.getElementById("element").setAttribute("data-underline",
"false");
    }
    function chipOnCreate() {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.element.querySelector('.e-editable-value').innerHTML =
chipCreation(editObj.value);
    }
    function chipCreation(data, isSuccess) {
        var value = '<div class="e-chip-list">';
        [].slice.call(data).forEach((val) => {
            value += '<div class="e-chip"> <span class="e-chip-text' +
(!isSuccess ? 'e-highlight' : '') + '> ' + val + '</span></div>';
        });
    }
</script>
```

```
        value += '</div>';  
        return value;  
    }  
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller  
{  
    public class SubmitModel  
    {  
        public string Name { get; set; }  
        public string PrimaryKey { get; set; }  
        public string Value { get; set; }  
    }  
    public ActionResult Index()  
    {  
        var frameWorkList = new string[] { "Android", "JavaScript",  
"jQuery", "TypeScript", "Angular", "React", "Vue", "Ionic" };  
        ViewBag.model = new { mode = "Box", dataSource = frameWorkList,  
placeholder = "Select skill" };  
        ViewBag.value = new string[] { "JavaScript", "jQuery" };  
        return View();  
    }  
    public void UpdateData([FromBody] SubmitModel model)  
    {  
        // User can process data here  
    }  
}
```

The output will be as follows.

Select frameworks:

JavaScript

jQuery

See Also

- [Indicate the server actions in the editor](#)

Data Binding

The Syncfusion ASP.NET MVC controls load the data either from local data sources or remote data services using the `dataSource` property and it supports the data type of an array or `DataManager`. Also supports different kind of data services such as OData, OData V4, Web API, and data formats such as XML, JSON, JSONP with the help of `DataManager` adaptors.

Local

To bind local data to the Syncfusion ASP.NET MVC controls, you can assign a JavaScript array of object or string to the `dataSource` property. The local data source can also be provided as an instance of the `DataManager`.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <span class="content-title"> Select customer name: </span>

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Type(Syncfusion
.EJ2.InPlaceEditor.InputType.DropDownList).Value(ViewBag.value).Model(ViewBa
g.model).Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80px;
    }
    #element {
        width: 150px;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    List<gameList> game = new List<gameList>();
    public ActionResult Index()
    {
        game.Add(new gameList { Id = "1", Name = "Maria Anders" });
        game.Add(new gameList { Id = "2", Name = "Ana Trujillo" });
        game.Add(new gameList { Id = "3", Name = "Antonio Moreno" });
        game.Add(new gameList { Id = "4", Name = "Thomas Hardy" });
        game.Add(new gameList { Id = "5", Name = "Chiristina Berglund" });
        game.Add(new gameList { Id = "6", Name = "Hanna Moos" });
        ViewBag.value = "Maria Anders";
        ViewBag.model = new { dataSource = game, fields = new { text =
"Name" }, Placeholder = "Select a customer" };
        return View();
    }
    public class gameList
    {
        public string Id { get; set; }
        public string Name { get; set; }
    }
}

```

The output will be as follows.

Select customer name: Maria Anders

Remote

To bind remote data to the Syncfusion ASP.NET MVC control, assign service data as an instance of **DataManager** to the **dataSource** property. To interact with remote data source, provide the endpoint URL.

OData V4

The OData V4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData V4 services. To fetch data from the server by using **DataManager** with the adaptor property configure as **ODataV4Adaptor**.

In the following sample, In-place Editor renders a **DropDownList** control and its **dataSource** property configured for fetching a data from the server by using **ODataV4Adaptor** with **DataManager**.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <span class="content-title"> Select customer name: </span>

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Type(Syncfusion
.EJ2.InPlaceEditor.InputType.DropDownList).Value(ViewBag.value).Model(ViewBa
g.model).Created("created").Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80px;
    }

    #element {
        width: 150px;
    }

    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>
<script>
    function created() {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.model.dataSource = new ej.data.DataManager({
            url: "https://services.odata.org/V4/Northwind/Northwind.svc/",
            adaptor: new ej.data.ODataV4Adaptor,
            crossDomain: true
        });
    };
</script>
```

```

        editObj.model.query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    }
</script>

```

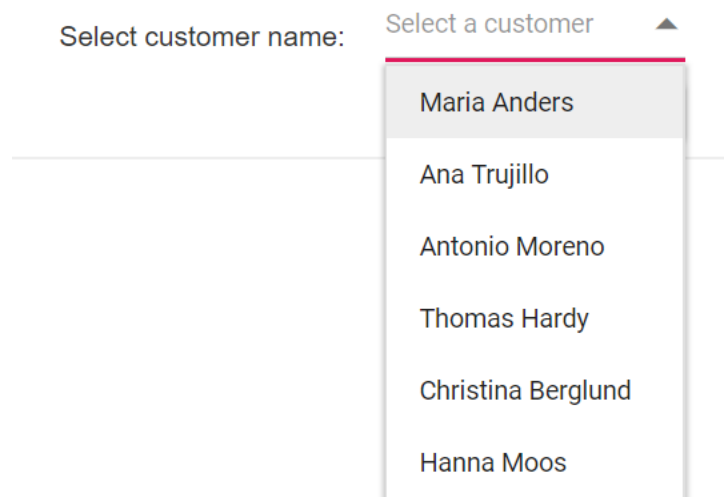
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new
        {
            placeholder = "Select a customer",
            fields = new { text = "ContactName", value = "CustomerID" }
        };
        ViewBag.value = "Maria Anders";
        return View();
    }
}

```

The output will be as follows.



Web API

Data can fetch from the server by using **DataManager** with the adaptor property configure as **WebApiAdaptor**.

In the following sample, In-place Editor render a **DropDownList** control and its **dataSource** property configured for fetching a data from the server by using **WebApiAdaptor** with **DataManager**.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <span class="content-title"> Select customer name: </span>

```

```

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Type(Syncfusion
.EJ2.InPlaceEditor.InputType.DropDownList).Value(ViewBag.value).Model(ViewBa
g.model).Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80px;
    }
    #element {
        width: 150px;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>
<script>
    new ej.data.DataManager({
        url:
'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Customers/',
        adaptor: new ej.data.WebApiAdaptor
    }).executeQuery(new ej.data.Query().take(8)).then((e) => {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.model.dataSource = e.result.d;
    });
</script>

```

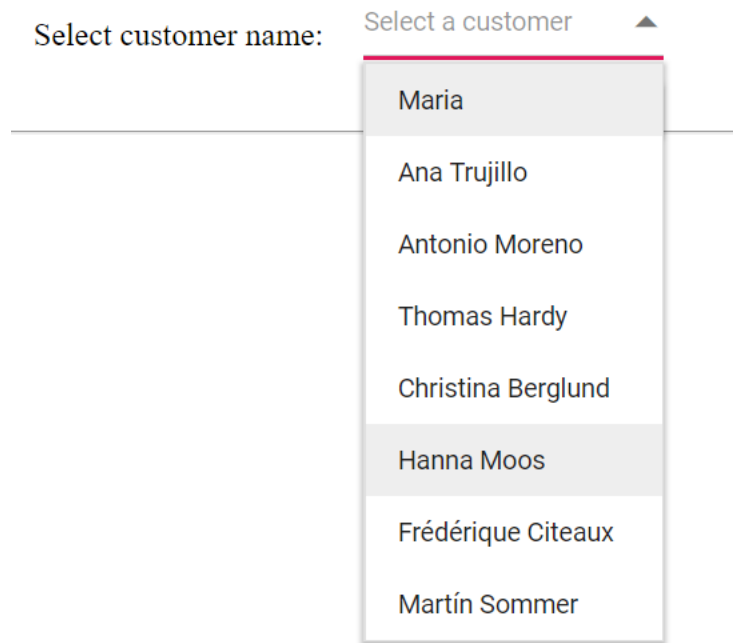
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new { placeholder = "Select a customer", fields =
new { text = "ContactName", value = "CustomerID" } };
        ViewBag.value = "Maria Anders";
        return View();
    }
}

```

The output will be as follows.



Integrate HTML5 controls (Template)

The In-place Editor supports adding HTML5 input controls using the [Template](#) property. The **Template** property can be given as either a **string** or a **query selector**.

As a string

The HTML element tag can be given as a string for the [Template](#) property. Here, the input is rendered as an HTML template.

```
`typescript
```

```
template: "<div><input type='text' id='name'></input></div>"
```

```
,
```

As a selector

The [Template](#) property also allows getting template content through query **selector**. Here, the input wrapper element 'ID' attribute is specified in the template.

```
`typescript
```

```
template: "#date"
```

```
,
```

Template mode, the [Value](#) property not handled by the In-place Editor control. So, before sending a value to the server, you need to modify at [ActionBegin](#) event, otherwise, an empty string will pass. In the following template sample, before submitting a data to the server, event argument and **Value** property content updated in the **ActionBegin** event handler.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <span class="content-title"> Select date: </span>
```



```

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Template("#date")
.Value(ViewBag.dateValue).ActionBegin("actionBegin").Render()
</div>
<div id="html-template" style="display: none">
    <input id="date" value="2018-05-23" type="date">
</div>

<style>
    #container {
        display: flex;
        justify-content: center;
    }
    #element {
        width: 150px;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>
<script>
    function actionBegin(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        var value = editObj.element.querySelector('#date').value;
        editObj.value = value;
        e.value = value;
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.dateValue = "2018-05-23";
        return View();
    }
}

```

The output will be as follows.

Select date: 2018-05-23

Validation

In-place Editor control supports validation and it can be achieved by adding rules to the [ValidationRules](#) property, its child property **key** must be same as **name** property, otherwise validation not performed. Submitting data to the server or calling the **validate** method validation executed.

Note: In-place editor is not a form component and it does not support `ejs-for` because the component has embedded into many other components like calendar, textboxes, etc.

Validation Rules

In-place Editor has following validation rules, which are used to perform validation.

Rules	Description	Example
----- ----- -----		
required	The input element must have any input values	a or 1 or -
email	The input element must have valid email format values	<inplace@syncfusion.com>
url	The input element must have valid url format values	http://syncfusion.com/
date	The input element must have valid date format values	12/25/2019
dateIso	The input element must have valid dateIso format values	2019-12-25
number	The input element must have valid number format values	1.0 or 1
maxLength	Input value must have less than or equal to maxLength character length	if maxLength : 5, [check] is valid and [checking] is invalid
minLength	Input value must have less than or equal to minLength character length	if minLength : 5, [testing] is valid and [test] is invalid
rangeLength	Input value must have value between rangeLength character length	if rangeLength : [4,5], [test] is valid and [key] is invalid
range	Input value must have value between range number	if range : [4,5], [4] is valid and [6] is invalid
max	Input value must have less than or equal to max number	if max : 3, [3] is valid and [4] is invalid
min	Input value must have less than or equal to min number	if min : 4, [5] is valid and [2] is invalid

CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the In-place Editor text

Use the following CSS to customize the default In-place Editor's text content properties like font-family, font-size, color and border bottom.

```
`CSS
/ To change color, font family and font size /
.e-inplaceeditor .e-editable-value-wrapper .e-editable-value {
border-bottom: 2px dotted green;
color: red;
font-size: 12px;
```

font-family: Segoe UI

}

,

Customizing the In-place Editor action buttons

Use the following CSS to customize the default In-place Editor's action buttons.

`CSS

/ To change icon color for save button /

```
.e-inplaceeditor .e-editable-action-buttons .e-btn-save.e-icon-btn .e-btn-icon.e-icons,
.e-inplaceeditor-tip .e-editable-action-buttons .e-btn-save.e-icon-btn .e-btn-icon.e-icons{
color: green;
}
```

/ To change icon color for cancel button /

```
.e-inplaceeditor .e-editable-action-buttons .e-btn-cancel.e-icon-btn .e-btn-icon.e-icons, .e-
inplaceeditor-tip .e-editable-action-buttons .e-btn-cancel.e-icon-btn .e-btn-icon.e-icons {
color: red;
}
```

/ To change background color for save button /

```
.e-inplaceeditor .e-editable-action-buttons .e-btn-save.e-icon-btn,
.e-inplaceeditor-tip .e-editable-action-buttons .e-btn-save.e-icon-btn {
background-color: antiquewhite;
}
```

/ To change background color for cancel button /

```
.e-inplaceeditor .e-editable-action-buttons .e-btn-cancel.e-icon-btn,
.e-inplaceeditor-tip .e-editable-action-buttons .e-btn-cancel.e-icon-btn {
background-color: antiquewhite;
}
```

,

Globalization

Localization

Localization library allows you to localize the default text content of the In-place Editor to different cultures using the [Locale](#) property. In-place Editor following keys will be localize based on culture.

| Locale key | en-US (default) |

|-----|-----|

| save | Close |

| cancel | Cancel |

| loadingText | Loading... |

| editIcon | Click to edit |

| editAreaClick | Click to edit |

| editAreaDoubleClick | Double click to edit |

To load translation object in an application use `load` function of `L10n` class. In the following sample, `French` culture is set to In-place Editor and change the tooltip text.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> EditableOn: </td>
            <td>
                @Html.EJS().DropDownList("dropDown").Width("auto").DataSource(ViewBag.editableOnData).Value(ViewBag.value).Change("onChange").Placeholder("Select edit type").Render()
            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">
                @Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.inplaceValue).Locale("fr-BE").Model(ViewBag.model).Render()
            </td>
        </tr>
    </table>
</div>
<style>
    .table-section {
        margin: 0 auto;
    }
    tr td:first-child {
        text-align: right;
        padding-right: 20px;
    }
    .sample-td {
        padding-top: 10px;
        min-width: 230px;
        height: 100px;
    }
</style>
<script>
    ej.base.L10n.load({
        'fr-BE': {
            'inplace-editor': {
                'save': 'enregistrer',
                'cancel': 'Annuler',
                'loadingText': 'Chargement...',
            }
        }
    });
</script>
```

```

        'editIcon': 'Cliquez pour éditer',
        'editAreaClick': 'Cliquez pour éditer',
        'editAreaDoubleClick': 'Double-cliquez pour éditer'
    }
}
});
function onChange(e) {
    var editObj = document.getElementById('element').ej2_instances[0];
    var editType = e.itemData.value;
    editObj.editableOn = editType;
    editObj.dataBind();
}
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new { placeholder = "Enter some text" };
        ViewBag.editableOnData = new string[] { "Click", "DbClick",
        "EditIconClick" };
        ViewBag.value = "Click";
        ViewBag.inplaceValue = "Andrew";
        return View();
    }
}

```

The output will be as follows.

EditableOn: Click ▼

Enter your name: Andrew 

Cliquez pour éditer

Right to left

Specifies the direction of the In-place Editor control using the [EnableRtl](#) property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

Note: It will not change based on the locale property.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <span class="content-title"> Enter your name: </span>

```

```
@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.v
alue).EnableRtl(true).Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80px;
    }
    #element {
        width: 150px;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.value = "Andrew";
        return View();
    }
}
```

The output will be as follows.

Enter your name:

Format

Formatting is a way of representing the value in different format. You can format the following mentioned controls with its `format` property, when it passed through the In-place Editor [Model](#) property.

- DatePicker
- DateRangePicker
- DateTimePicker
- NumericTextBox
- Slider
- TimePicker

CSHTML

```

<div id='container'>
    <span class="content-title"> Select date: </span>

@Html.EJS().InPlaceEditor("element").Type(Syncfusion.EJ2.InPlaceEditor.Input
Type.Date).Value(ViewBag.value).Model(ViewBag.model).Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
</style>

```

CONTROLLER.CS

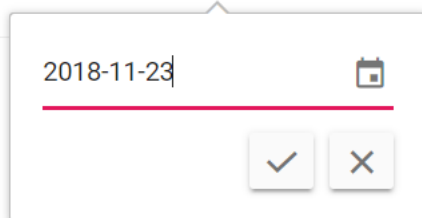
```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.value = new DateTime(2018, 11, 23);
        ViewBag.model = new { placeholder = "Select date", format = "yyyy-
MM-dd" };
        return View();
    }
}

```

The output will be as follows.

Select date: 11/23/2018

**How To****Dynamically move input to edit mode**

At control initial load, if you want to open editor state without interacting In-place Editor input element, it can be achieved by configuring the [EnableEditMode](#) property to `true`.

In the following sample, editor opened at initial load and when toggling a checkbox, it will remove or open the editor.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> Disabled: </td>
            <td>
                @Html.EJS().CheckBox("enable").Change("onChange").Label("Disable").Checked(false).Render()
            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">
                @Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.value).Model(ViewBag.model).Render()
            </td>
        </tr>
    </table>
</div>
<style>
    .table-section {
        margin: 0 auto;
    }
    tr td:first-child {
        text-align: right;
        padding-right: 20px;
    }
    .sample-td {
        padding-top: 10px;
        min-width: 230px;
        height: 100px;
    }
</style>
<script>
    function onChange(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.disabled = e.checked;
        editObj.dataBind();
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new { placeholder = "Enter some text" };
        ViewBag.value = "Andrew";
        return View();
    }
}

```


The output will be as follows.

Disabled: ☒ Enable

Enter your name: ✕

✓ ✕

Disable the edit mode specifically

The edit mode of In-place Editor can be disabled by setting the [Disabled](#) property value to `true`. In the following sample, when check or uncheck the checkbox, In-place Editor control will disable or enable the edit mode.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> Disabled: </td>
            <td>

@Html.EJS().CheckBox("enable").Change("onChange").Label("Disable").Checked(false).Render()

            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.value).Model(ViewBag.model).Render()

            </td>
        </tr>
    </table>
</div>
<style>

#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.table-section {
    margin: 0 auto;
}
tr td:first-child {
    text-align: right;
```

```

padding-right: 20px;
}
.sample-td {
padding-top: 10px;
min-width: 230px;
height: 100px;
}
</style>
<script>
function onChange(e) {
    var editObj = document.getElementById('element').ej2_instances[0];
    editObj.disabled = e.checked;
    editObj.dataBind();
}
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new { placeholder = "Enter some text" };
        ViewBag.value = "Andrew";
        return View();
    }
}

```

The output will be as follows.

Disabled: ☒ Disable

Enter your name:

Add custom indication to unsaved value

You can add custom indication to unsaved input value by using the [ActionSuccess](#) event, when data not submitted to the server.

In this sample, the `ActionSuccess` event configured and the `Url` property not included. Then submit button clicked, the current editor value saved into input and data sending to server action prevented due to the `Url` property not configured.

But `ActionSuccess` event will trigger the handler function with `null` argument values. In handler function data property `PrimaryKey` value checked, whether it empty or not. If it is empty custom class, added in the `e-value-wrapper` element to customize its styles.

Note: To send input value to local, set the `Url` property as empty.

CSHTML

```

@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <span class="content-title"> Enter your name: </span>
    <div id='element'></div>

@Html.EJS().InPlaceEditor("element").Mode(RenderMode.Inline).Value(ViewBag.v
alue).Model(ViewBag.model).ActionSuccess("actionSuccess").Render()
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80px;
    }
    #element {
        width: 150px;
    }
    .content-title {
        font-weight: 500;
        margin-right: 20px;
        display: flex;
        align-items: center;
    }
    .e-inplaceeditor .e-editable-value-wrapper .e-editable-value.e-send-
error {
        color: red;
    }
</style>
<script>
    function actionSuccess(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        var pk = e.data['PrimaryKey'];
        if (ej.base.isNullOrUndefined(pk) || pk === '') {
            document.querySelector('.e-editable-value').classList.add('e-
send-error');
        }
    }
</script>

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new { placeholder = "Enter some text" };
        ViewBag.value = "Andrew";
        return View();
    }
}

```

The output will be as follows.

Enter your name: Andrew

Set custom animation for popup mode

In popup mode, the In-place Editor rendered with the Syncfusion ASP.NET MVC **Tooltip** control. You can use tooltip properties and events to customize the popup by configure properties into the [Model](#) property inside the [PopupSettings](#) API.

In the following sample, popup animation can be customized by passing animation effect using the **Model** property and the dynamic animation effect changes configured from the Syncfusion ASP.NET MVC **DropDownList** control **change** event.

CSHTML

```
@using Syncfusion.EJ2.InPlaceEditor;
<div id='container'>
    <table class="table-section">
        <tr>
            <td> Open Animation: </td>
            <td>
                @Html.EJS().DropDownList("openDropDown").DataSource(ViewBag.openAnimateData)
                .Value(ViewBag.textValue).Change("onChange").Placeholder("Select a animate
                type").PopupHeight("150px").Render()
            </td>
        </tr>
        <tr>
            <td class="sample-td"> Enter your name: </td>
            <td class="sample-td">
                @Html.EJS().InPlaceEditor("element").Mode(RenderMode.Popup).Value(ViewBag.va
                lue).PopupSettings(ViewBag.popupSettings).Model(ViewBag.model).Render()
            </td>
        </tr>
    </table>
</div>
<style>
    #container {
        display: flex;
        justify-content: center;
    }
    .table-section {
        margin: 0 auto;
    }
    tr td:first-child {
        text-align: right;
        padding-right: 20px;
    }
    .sample-td {
        padding-top: 10px;
        min-width: 230px;
        height: 100px;
    }
</style>
```

```
<script>
    function onChange(e) {
        var editObj = document.getElementById('element').ej2_instances[0];
        editObj.popupSettings.model.animation.open.effect = e.value;
        editObj.dataBind();
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.model = new { placeholder = "Enter some text" };
        ViewBag.openAnimateData = new string[] { "None", "FadeIn",
        "FadeZoomIn", "ZoomIn" };
        ViewBag.textValue = "ZoomIn";
        ViewBag.value = "Andrew";
        ViewBag.popupSettings = new
        Syncfusion.EJ2.InPlaceEditor.InPlaceEditorPopupSettings { Model = new {
        animation = new { open = new { effect = "ZoomIn", duration = 1000, delay = 0
        } } } };
        return View();
    }
}
```

The output will be as follows.

Open Animation:

Enter your name:

Image Editor

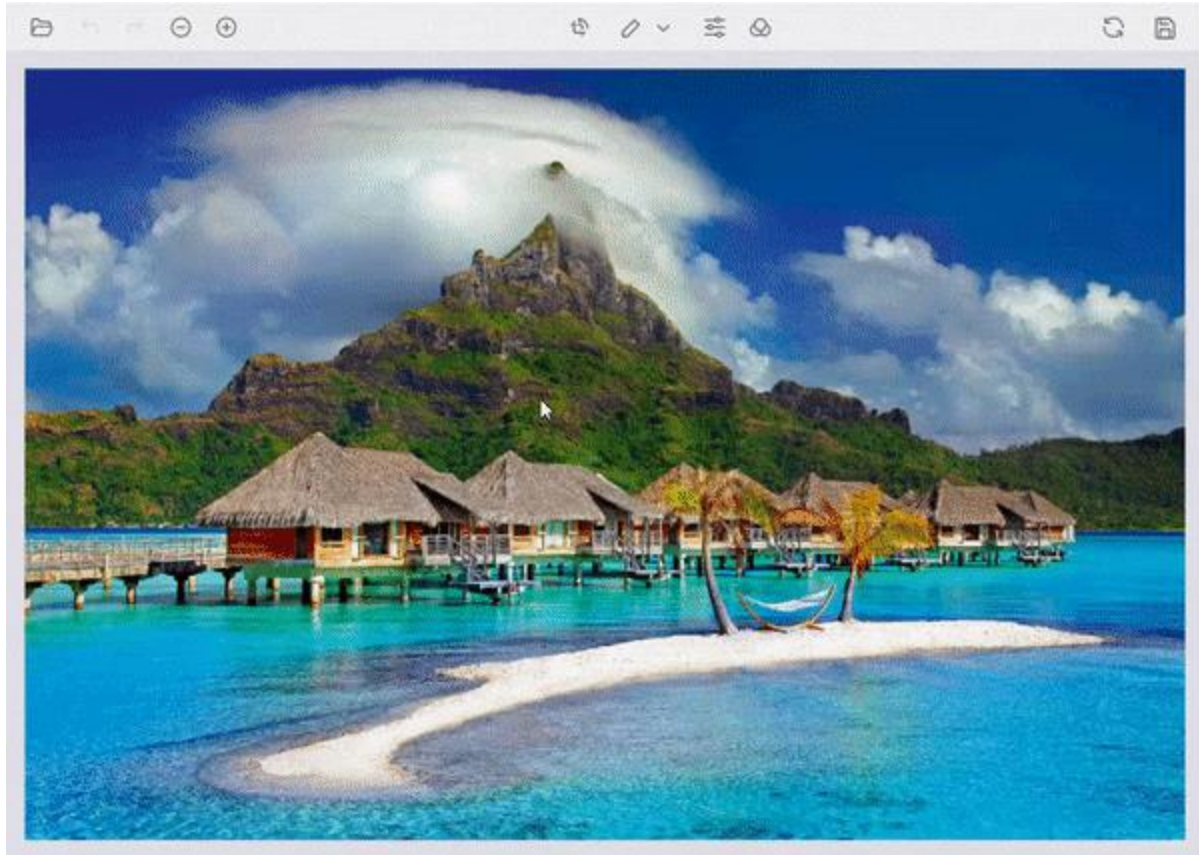
End User Capabilities in the ASP.NET MVC Image Editor control

The following operations are available for end-users and the same is explained briefly in these sections.

Open an image

To open an image in the image editor, do the following steps.

- Click the Open icon from the left side of the toolbar.
- The file explorer lists only JPEG, PNG, JPG format files.
- Select the image from the list of the images from the file explorer window.



Zooming

Image zooming can be performed in the following ways.

- Using toolbar
- Using pinch zoom in touch enabled devices
- Using mouse wheel
- Using keyboard

Using toolbar

To zoom in or out the image in the image editor, do the following steps.

- The Zoom In/ Out option only enabled after opening the image.

Using pinch

To zoom in or out the image in the image editor, do the following steps.

- Touch with two fingers to perform zooming.
- Zoom in and out controlled by touch gestures.

Using Mouse wheel

To zoom in or out the image in the image editor, do the following steps.

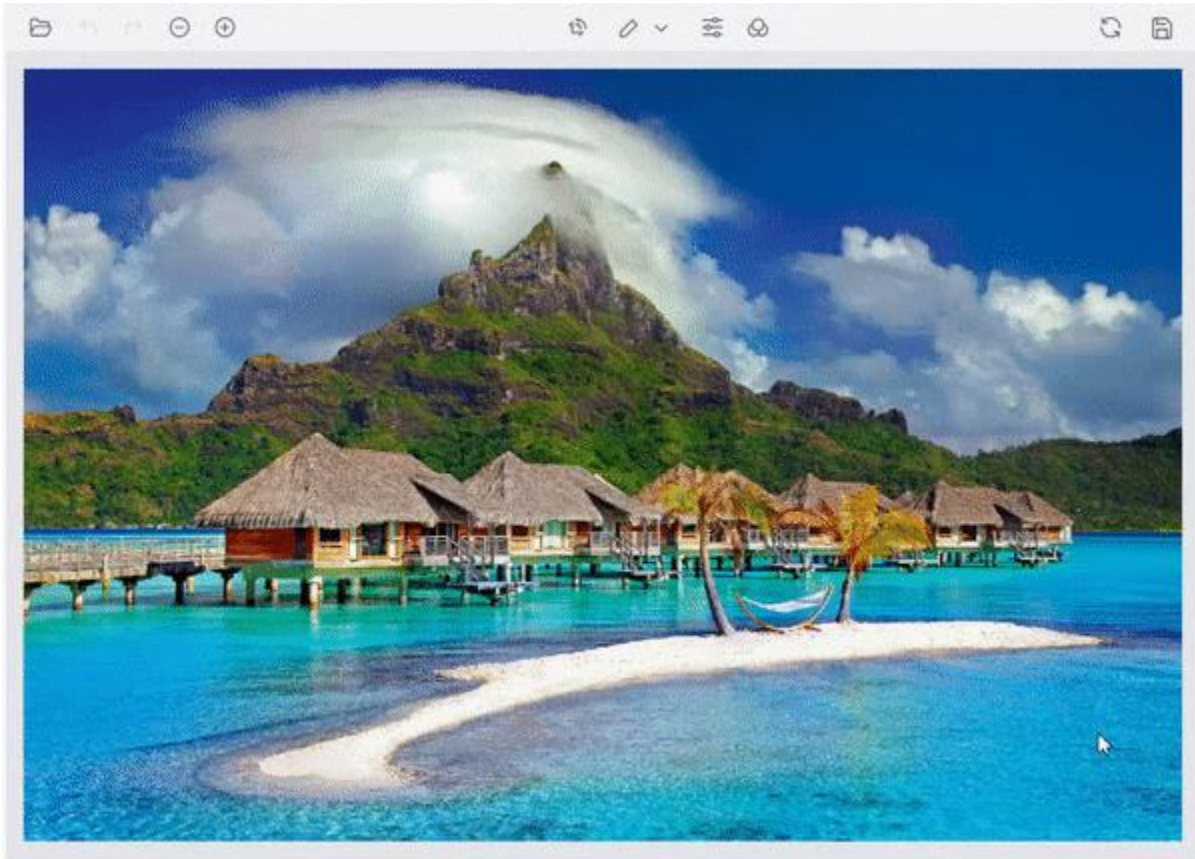
- Press the ctrl key and scroll the mouse wheel to perform zooming.

- The zoom in and out controlled by the mouse wheel.

Using keyboard

To zoom in or out the image in the image editor, do the following steps.

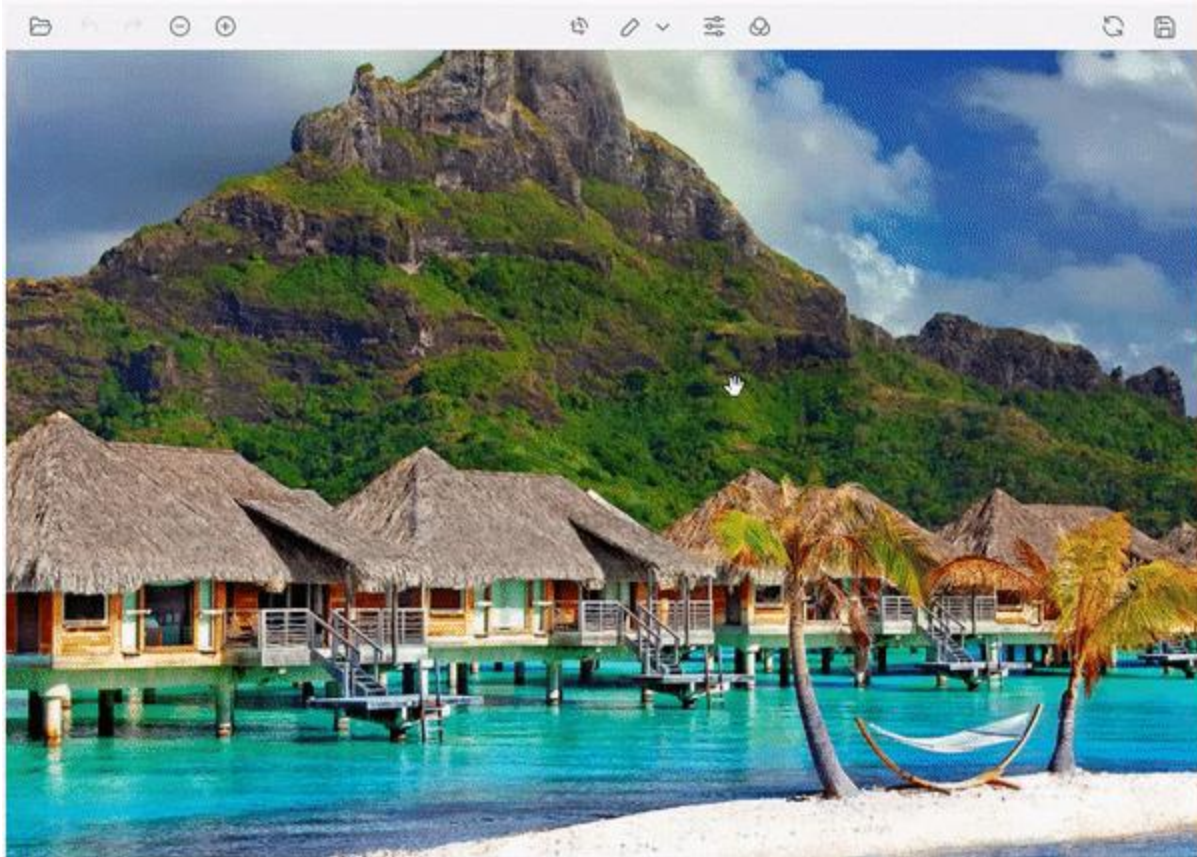
- Press the ctrl key with '+' button from the keyboard to zoom in an image.
- Press the ctrl key with '-' button from the keyboard to zoom out an image.



Panning

To pan an image in the image editor, do the following steps.

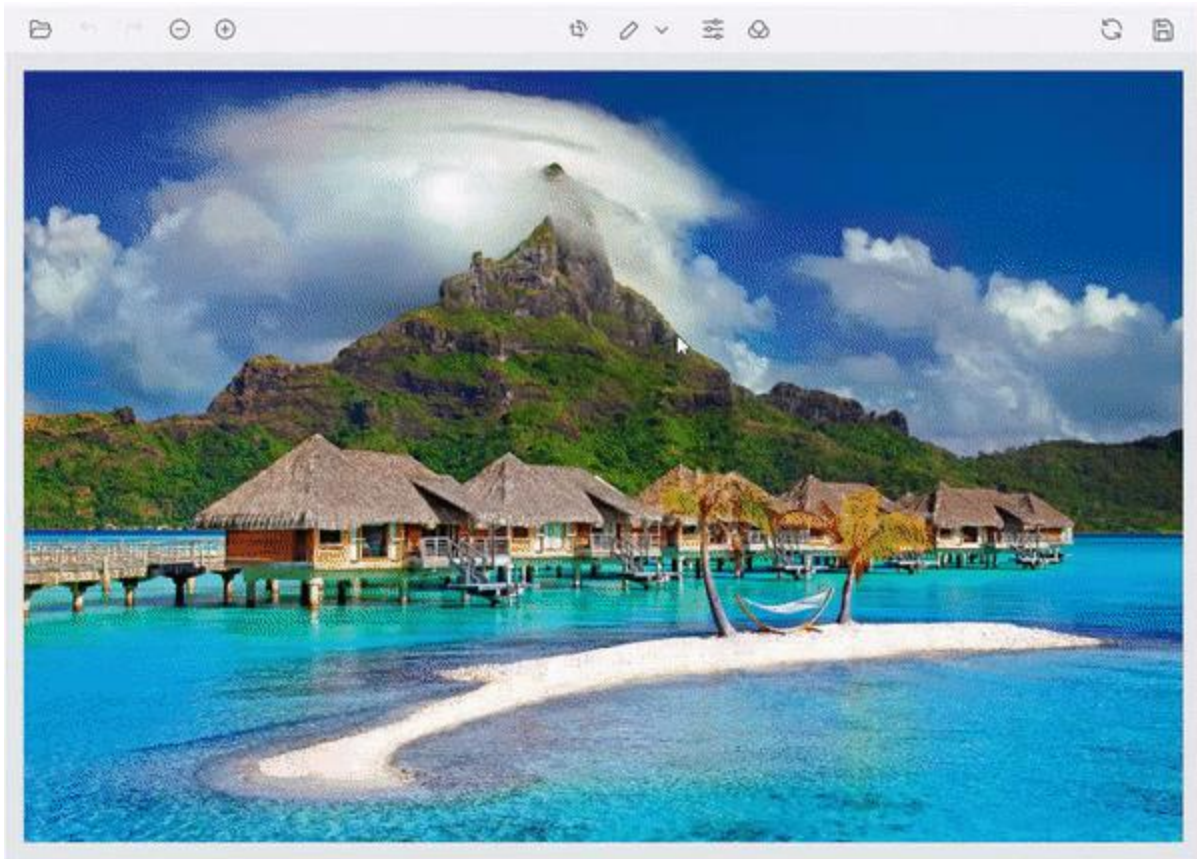
- Click on the image and do dragging to move or pan the image.
- Panning option will be enabled in the following two cases.
- If the selection is applied for cropping an image.
- If the image size exceeds the canvas size while zooming an image.



Cropping and image transformation

To crop an image in the image editor, do the following steps.

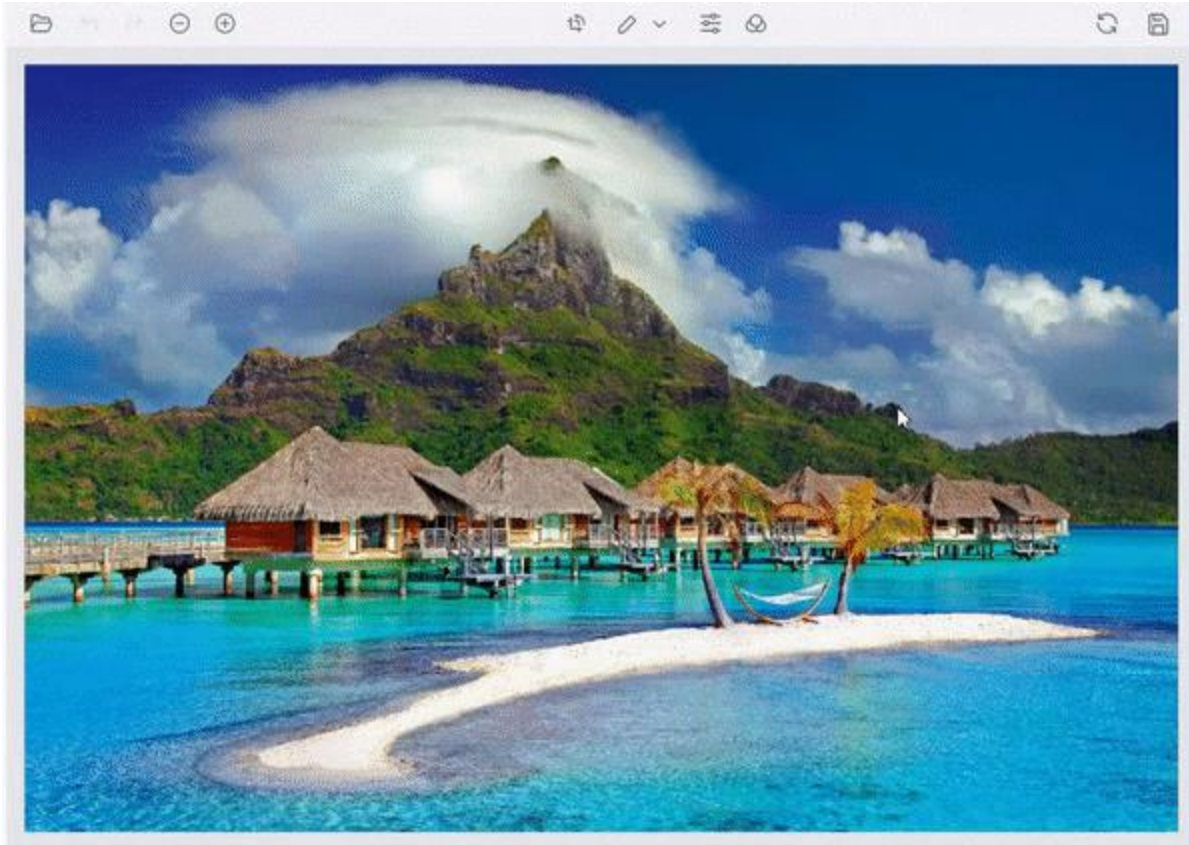
- Cropping can be performed based on the selection in an image editor.
- To perform selection, click the crop button in the toolbar which opens the contextual toolbar that shows crop selection options, rotate options, and flip options.
- Click the crop selection button and select the type of selection such as custom, circle, square, and ratio selection from the popup.
- Once selection is completed, do panning to move the image to get the cropped region.
- Utilize the rotate or flip buttons to execute the image transformation, including any inserted annotations.
- Once the cropping region is finalized in the image click the tick icon at the top right of the toolbar to crop the image.



Annotations

To add annotations to an image in the image editor, do the following steps.

- To add annotation, click the annotation button in the toolbar and select the type of annotations such as Line, Rectangle, Ellipse, Path, Arrow, Text, or Freehand drawing to be inserted to the image editor.
- Once the annotation is added to the image, that can be repositioned by clicking and dragging the annotations using mouse as well as resized by clicking and resizing the selection circle to be placed around the annotations.
- To rotate annotations, you can simply grab the circle located at the bottom of the annotation. The rotation can be applicable to all the annotations except text annotation.
- Customize the annotations by changing their color, stroke width, font family, and font size through the contextual toolbar. The contextual toolbar will be enabled whenever the annotations are selected.
- When annotations are selected in the Image Editor, the quick access toolbar becomes active, providing convenient access to various actions such as duplicating, deleting, or editing text associated with the selected annotation. This toolbar enables users to perform these common operations quickly and efficiently, streamlining their workflow and enhancing the overall editing experience.



Filtering and fine-tune

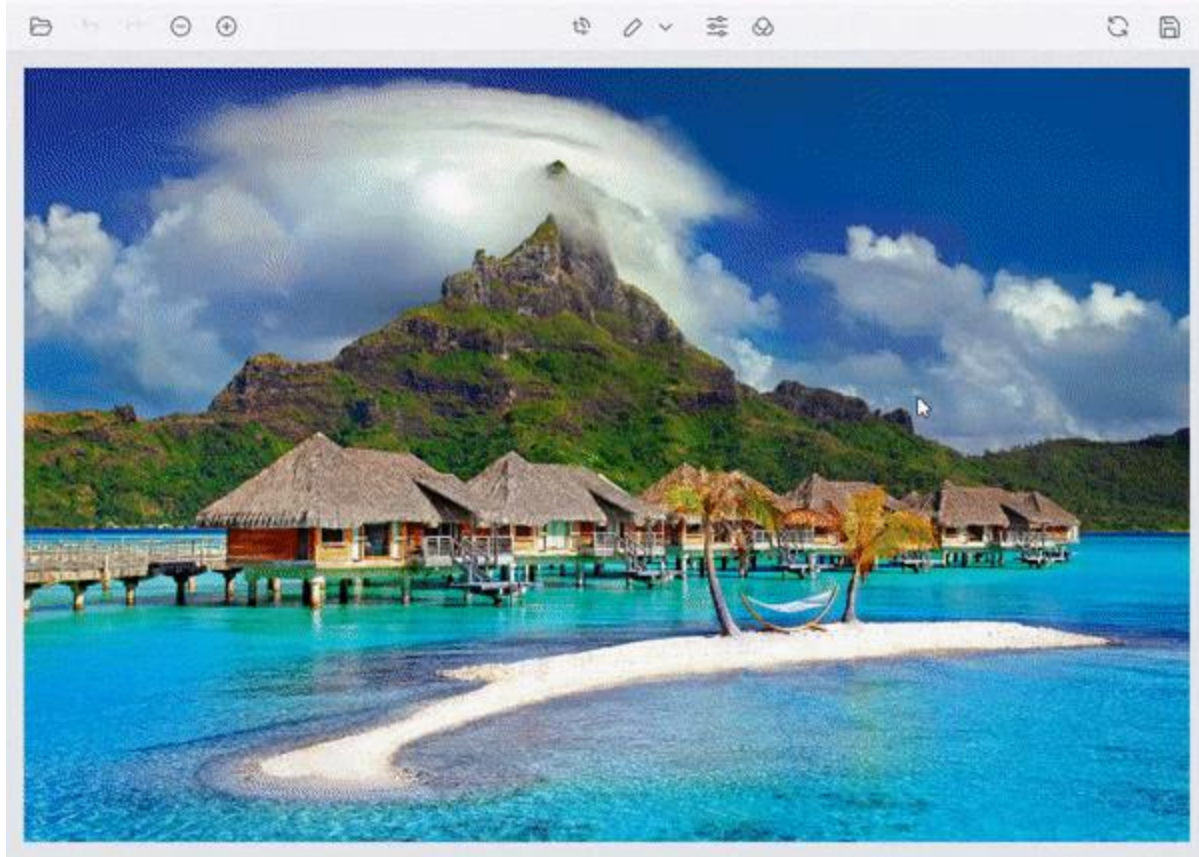
To perform fine-tuning on an image in the image editor, do the following steps.

- Click the fine-tune button which displays the list of fine-tuning available in the image editor.
- Click one of the fine-tune options from the list of options which shows a slider to adjust the corresponding filter.
- Click on the canvas or tick icon at the right corner of the toolbar in the image editor to apply the modifications.

To apply filters on an image in the image editor, do the following steps.

- Click the filter button which displays the list of filters available in the image editor.
- Click the filter from list of options to apply the corresponding filter to an image.
- Click on the canvas or tick icon at the right corner of the toolbar in the image editor to apply the modifications.

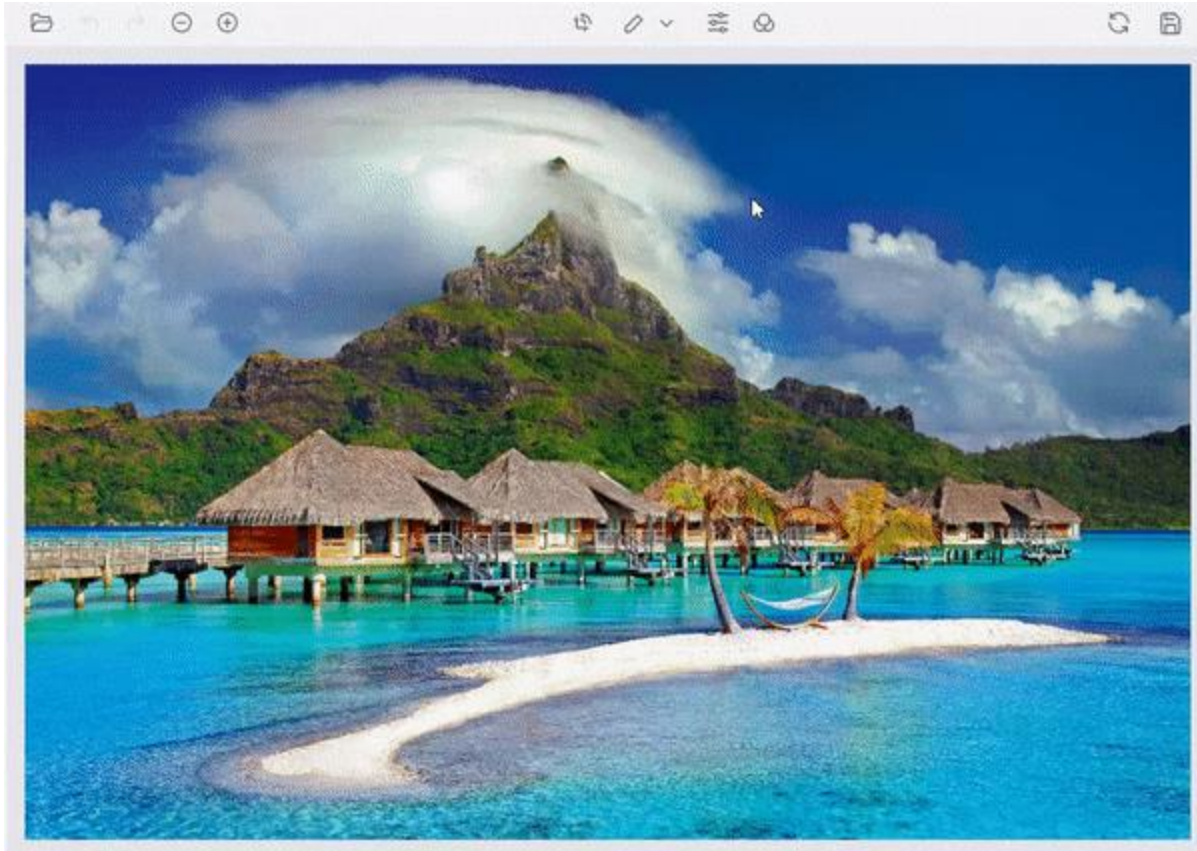
The Filters and Fine-tunes feature are not accessible within Safari due to compatibility limitations.



Undo and redo the operations

To undo and redo the actions performed in an image editor, do the following steps.

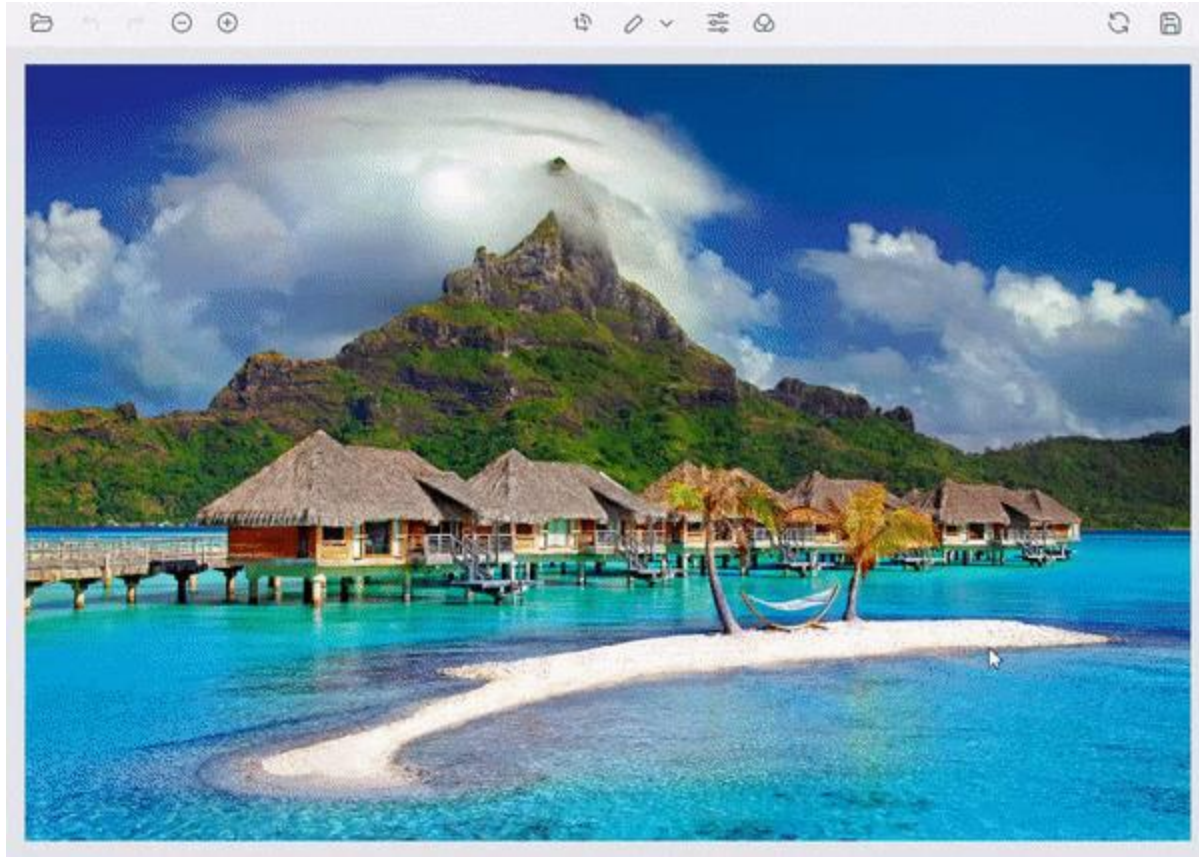
- The undo button will be enabled once the action is performed in an image editor.
- The redo button will be enabled once the undo action is performed in an image editor.
- Click the undo or redo button at the left side of the toolbar to perform undo and redo operation.
- Ctrl + Z and Ctrl + Y facilitates this process by allowing users to undo and redo actions, respectively.



Reset an image

To revert all the changes done in an image editor, do the following steps.

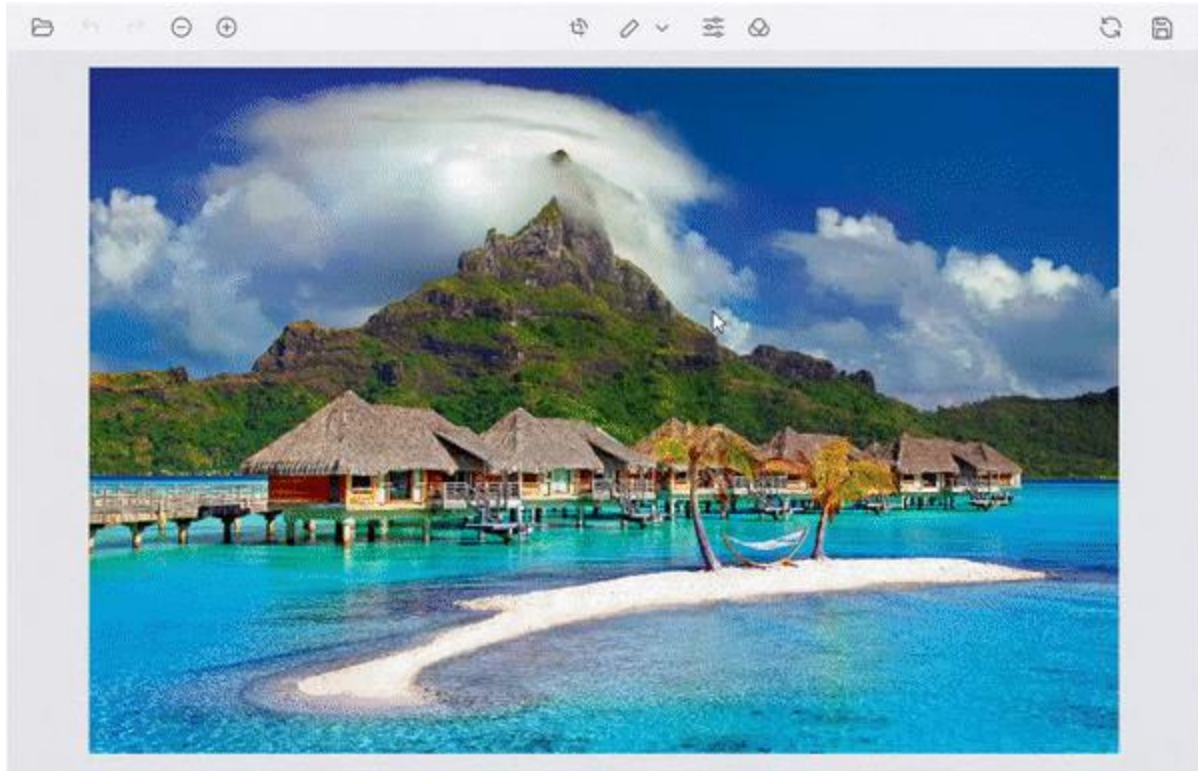
- Click the reset button which is located on the right side of the toolbar.
- This will revert all the changes performed in the image editor.



Export an image

To save the modified image in an image editor, do the following steps.

- Click the save button which is located on the right side of the toolbar.
- Ctrl + S facilitates this process by providing users with the ability to save the image.
- Select the type of file to be saved from the popup to save with current modification done in an image.



Getting Started with ASP.NET MVC Image Editor Control

This section briefly explains about how to include [ASP.NET MVC Image Editor](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Image Editor control

Now, add the Syncfusion ASP.NET MVC Image Editor control in **~/Views/Home/Index.cshtml** page.

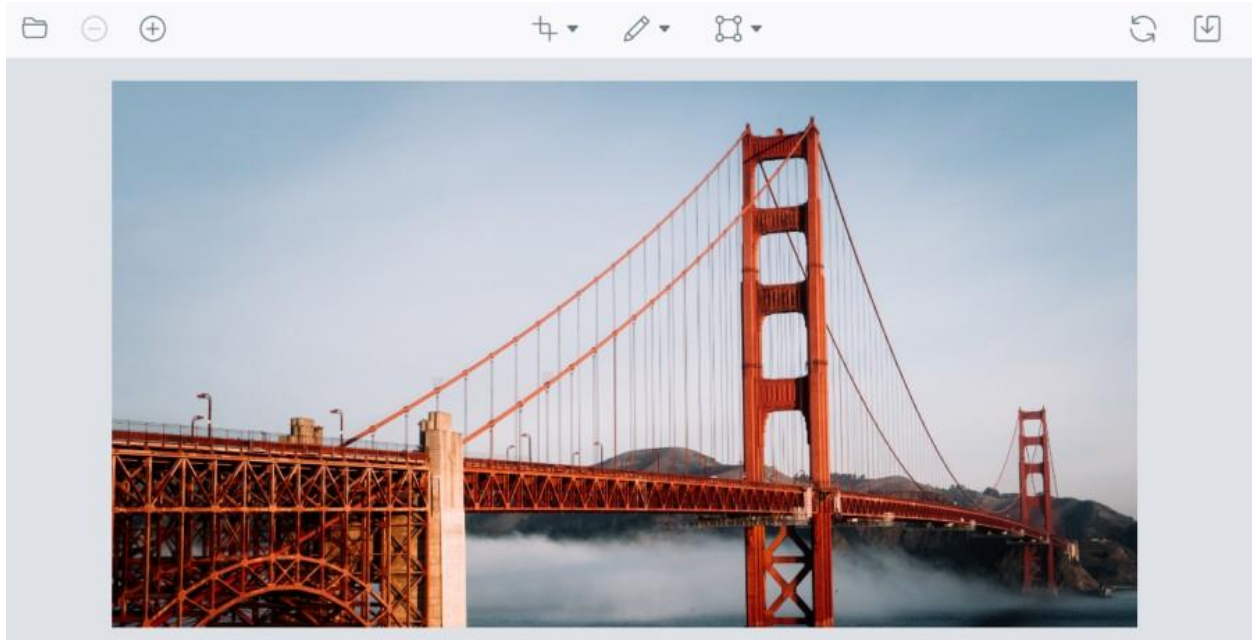
CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Render()
```



```
</div>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Image Editor control will be rendered in the default web browser.



Open and Save in the ASP.NET MVC Image Editor control

To import an image into the canvas, it must first be converted into a blob object. The Uploader component can be used to facilitate the process of uploading an image from the user interface. Once the image has been uploaded, it can then be converted into a blob and drawn onto the canvas.

To save an edited image in the Image Editor control, use the `toBlob` method to convert it to a blob object. This will save the image with any annotations or filters that have been applied during the editing process. The saved image can be stored as raw image data or as an image file.

Open

The **open** method in the Image Editor control offers the capability to open an image by providing it in different formats. This method accepts various types of arguments, such as a base64-encoded string, raw image data, or a hosted/online URL. You can pass either the file name or the actual image data as an argument to the **open** method, and it will load the specified image into the Image Editor control. This flexibility allows you to work with images from different sources and formats, making it easier to integrate and manipulate images within the Image Editor control.

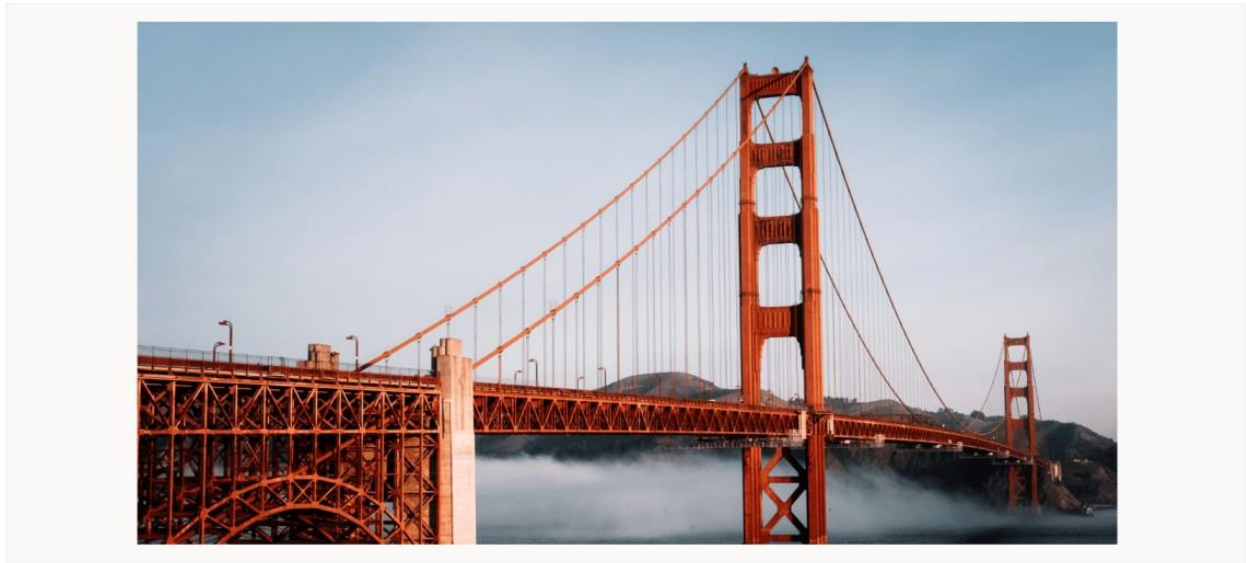
CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Supported image formats

The Image Editor control supports three common image formats: PNG, JPEG, and SVG. These formats allow you to work with a wide range of image files within the Image Editor.

When it comes to saving the edited image, the default file type is set as PNG. This means that when you save the edited image without specifying a different file type, it will be saved as a PNG file. However, it's important to note that the Image Editor typically provides options or methods to specify a different file type if desired. This allows you to save the edited image in formats other than the default PNG, such as JPEG or SVG, based on your specific requirements or preferences.

Save as image

The `export` method in the Image Editor control enables you to save the modified image as a file on the local device. This method accepts two parameters: the file name and the file type.

By providing a file name, you can specify the desired name for the saved image file. Additionally, you can also specify the file type to determine the format in which the image should be saved. This allows you to save the image according to your specific requirements or preferences, such as PNG, JPEG, or SVG.

By utilizing the `export` method with the appropriate file name and file type, you can conveniently save the modified image as a file on the local device, ensuring that it is easily accessible and shareable.

In the following example, the `export` method is used in the button click event.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
```

```
var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
    if (ej.base.Browser.isDevice) {
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
    } else {
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
    }
}
function clickHandler() {
    var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
    imageEditorObj.export("PNG", "Syncfusion"); // File type, file name
}
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    List<object> items = new List<object>();
    items.Add(new
    {
        text = "Png"
    });
    items.Add(new
    {
        text = "Jpeg"
    });
    items.Add(new
    {
        text = "Svg"
    });
    ViewBag.datasource = items;
    return View();
}
```

```
}
```

Output be like the below.



File opened event

The **FileOpened** event is triggered in the Image Editor control after an image is successfully loaded. It provides the **FileOpenEventArgs** as the event argument, which contains two specific arguments:

- **FileName**: This argument is a string that contains the file name of the opened image. It represents the name of the file that was selected or provided when loading the image into the Image Editor.
- **FileType**: This argument is a string that contains the type of the opened image. It specifies the format or file type of the image that was loaded, such as PNG, JPEG, or SVG.

By accessing these arguments within the **FileOpened** event handler, you can retrieve information about the loaded image, such as its file name and file type. This can be useful for performing additional actions or implementing logic based on the specific image that was opened in the Image Editor control.

Saving event

The **saving** event is triggered in the Image Editor control when an image is being saved to the local disk. It provides the **SaveEventArgs** as the event argument, which includes the following specific arguments:

- **FileName**: This argument is a string that holds the file name of the saved image. It represents the name of the file that will be used when saving the image to the local disk.
- **FileType**: This argument is a string indicating the type or format of the saved image. It specifies the desired file type in which the image will be saved, such as PNG, JPEG, or SVG.
- **Cancel**: This argument is a boolean value that can be set to true in order to cancel the saving action. By default, it is set to false, allowing the saving process to proceed. However, if you want to prevent the saving action from occurring, you can set **Cancel** to true within the event handler.

By accessing these arguments within the Saving event handler, you can retrieve information about the file name and file type of the image being saved. Additionally, you have the option to cancel the saving action if necessary.

Created event

The `created` event is triggered once the Image Editor control is created. This event serves as a notification that the component has been fully initialized and is ready to be used. It provides a convenient opportunity to render the Image Editor with a predefined set of initial settings, including the image, annotations, and transformations.

Destroyed event

The `destroyed` event is triggered once the Image Editor control is destroyed or removed from the application. This event serves as a notification that the component and its associated resources have been successfully cleaned up and are no longer active.

Reset an image

The `reset` method in the Image Editor control provides the capability to undo all the changes made to an image and revert it back to its original state. This method is particularly useful when multiple adjustments, annotations, or transformations have been applied to an image and you want to start over with the original, unmodified version of the image.

By invoking the `reset` method, any modifications or edits made to the image will be undone, and the image will be restored to its initial state. This allows you to easily discard any changes and begin again with the fresh, unaltered image.

Selection cropping in the ASP.NET MVC Image Editor control

The cropping feature in the Image Editor allows you to select and crop specific regions of an image. It offers different selection options, including custom shapes, squares, circles, and various aspect ratios such as 2:3, 3:2, 3:4, 4:3, 4:5, 5:4, 5:7, 7:5, 9:16, and 16:9.

To perform a selection, you can use the `select` method, which allows you to define the desired selection area within the image. Once the selection is made, you can then use the `crop` method to crop the image based on the selected region. This enables you to extract and focus on specific parts of the image while discarding the rest.

Insert custom / square / circle region

The `select` method allows to perform selection based on the type of selection. Here, the `select` method is used to perform the selection as custom, circle, or square. The selection region can also be customized using the `select` method based on the parameters below.

- `type` - Specify the type of selection
- `startX` - Specify the x-coordinate of the selection region's starting point
- `startY` - Specify the y-coordinate of the selection region's starting point
- `width` - Specify the width of the selection region
- `height` - Specify the height of the selection region

Here is an example of square selection using the `select` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
```

```

    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("customClick").CssClass("e-
primary").Content("Custom").Click("customClick").Render()
@Html.EJS().Button("squareClick").CssClass("e-
primary").Content("Square").Click("squareClick").Render()
@Html.EJS().Button("circleClick").CssClass("e-
primary").Content("Circle").Click("circleClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function customClick() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.select("Custom");
    }
    function squareClick() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.select("Square");
    }
    function circleClick() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.select("Circle");
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;

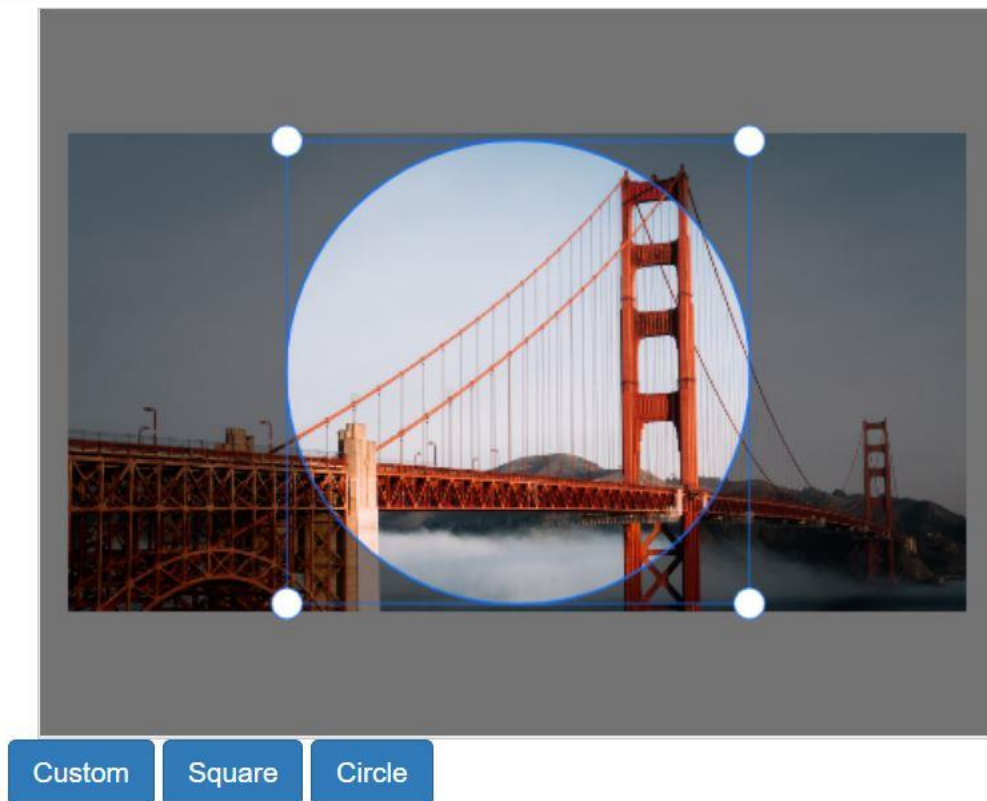
```

```
}  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Insert selection based on aspect ratio

The `select` method is used to perform the selection with the various aspect ratios such as 2:3, 3:2, 3:4, 4:3, 4:5, 5:4, 5:7, 7:5, 9:16, and 16:9. The selection region can also be customized using the `select` method based on the parameters below.

- type - Specify the type of selection
- startX - Specify the x-coordinate of the selection region's starting point
- startY - Specify the y-coordinate of the selection region's starting point

Here is an example of ratio selection using the `select` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
```

```

    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.select("16:9");
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Output be like the below.



Crop an image

The `crop` method allows cropping based on the selected region. Here is an example of cropping the selection region using the `crop` method.

Here is an example of circle cropping using the `select` and `crop` method.

CSHTML

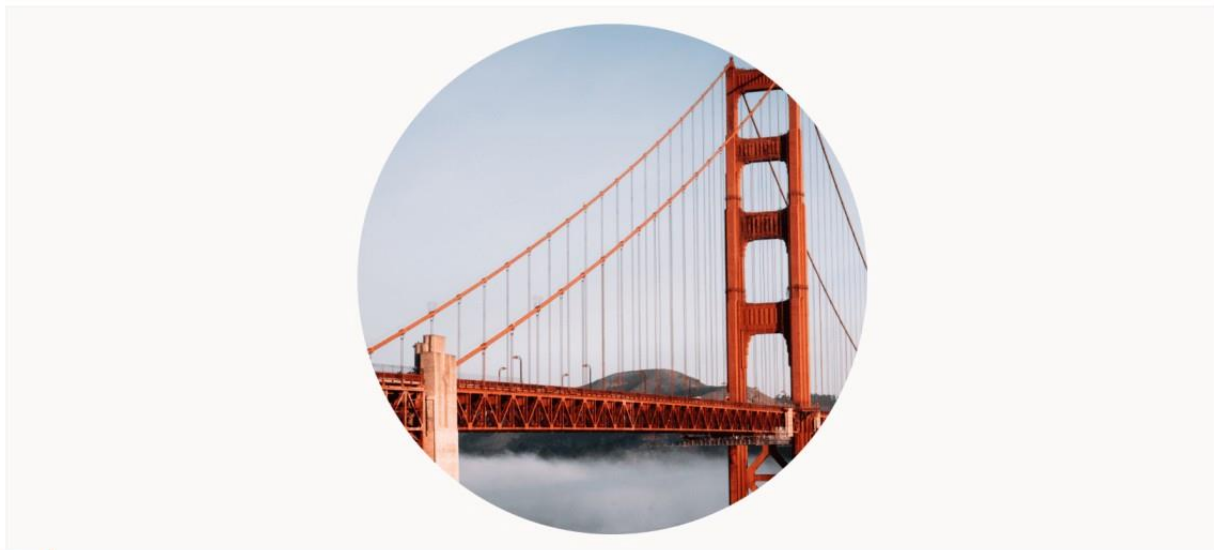
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.select("Circle");
        imageEditorObj.crop();
    }
</script>
<style>
    .image-editor {
```

```
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
}
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Click

Cropping event

The **cropping** event is triggered when performing cropping on the image. This event is passed an object that contains information about the cropping event, such as the start and end point of the selection region. And this event uses **CropEventArgs** to handle the cropping action in the image.

The parameter available in the **cropping** event is,

CroppingEventArgs.startPoint – The x and y coordinates of a start point as **ImageEditorPoint** of the selection region.

CroppingEventArgs endPoint - The x and y coordinates of an end point as ImageEditorPoint of the selection region.

CroppingEventArgs cancel - To cancel the cropping action.

Annotation in the ASP.NET MVC Image Editor control

The Image Editor allows adding annotations to the image, including text, freehand drawings, and shapes like rectangles, ellipses, arrows, paths, and lines. This gives the flexibility to mark up the image with notes, sketches, and other visual elements as needed. These annotation tools can help to communicate and share ideas more effectively.

Text annotation

The text annotation feature in the Image Editor provides the capability to add and customize labels, captions, and other text elements directly onto the image. With this feature, you can easily insert text at specific locations within the image and customize various aspects of the text to meet your requirements.

You have control over the customization options including text content, font family, font style and font size for the text annotation.

Add a text

The `drawText` method in the Image Editor allows you to insert a text annotation into the image with specific customization options. This method accepts the following parameters:

- `x`: Specifies the x-coordinate of the text, determining its horizontal position within the image.
- `y`: Specifies the y-coordinate of the text, determining its vertical position within the image.
- `text`: Specifies the actual text content to be added to the image.
- `fontFamily`: Specifies the font family of the text, allowing you to choose a specific typeface or style for the text.
- `fontSize`: Specifies the font size of the text, determining its relative size within the image.
- `bold`: Specifies whether the text should be displayed in bold style. Set to true for bold text, and false for regular text.
- `italic`: Specifies whether the text should be displayed in italic style. Set to true for italic text, and false for regular text.
- `color`: Specifies the font color of the text, allowing you to define the desired color using appropriate color values or names.

By utilizing the DrawText method with these parameters, you can precisely position and customize text annotations within the image. This provides the flexibility to add labels, captions, or other text elements with specific font styles, sizes, and colors, enhancing the visual presentation and clarity of the image.

Here is an example of adding a text in a button click using `drawText` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
```

```
var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
    if (ej.base.Browser.isDevice) {
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
    } else {
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.drawText(500, 500, 'Syncfusion', 'Arial', 100, true,
true, '#000');
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Multiline text

The `drawText` method in the Image Editor control is commonly used to insert text annotations into an image. If the provided text parameter contains a newline character (`\n`), the text will be automatically split into multiple lines, with each line appearing on a separate line in the annotation.

Here is an example of adding a multiline text in a button click using `drawText` method.

CSHTML

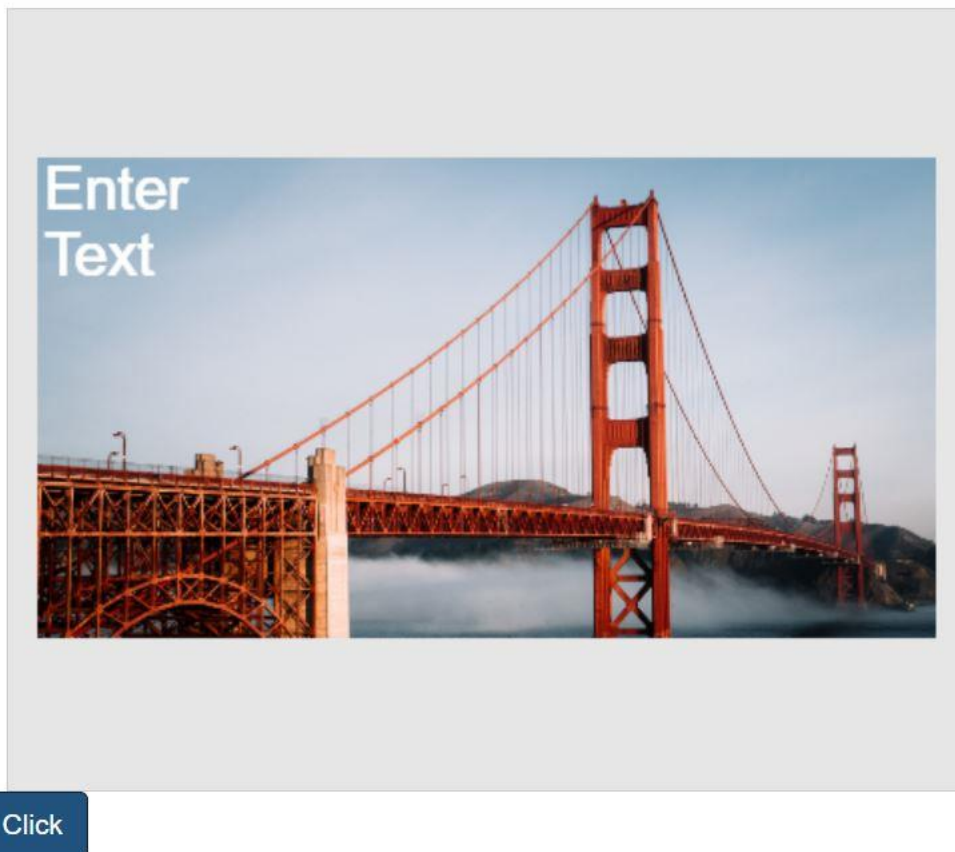
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
    string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        var dimension = imageEditorObj.getImageDimension();
        imageEditorObj.drawText(dimension.x, dimension.y, 'Enter\nText');
    }
</script>
<style>
```

```
.image-editor {  
    margin: 0 auto;  
}  
.e-img-editor-sample {  
    height: 80vh;  
    width: 100%;  
}  
@@media only screen and (max-width: 700px) {  
    .e-img-editor-sample {  
        height: 75vh;  
        width: 100%;  
    }  
}  
.control-wrapper {  
    height: 100%;  
}  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Delete a text

The `deleteShape` method in the Image Editor control allows you to remove a text annotation from the image editor. To use this method, you need to pass the `shapeld` of the annotation as a parameter.

The `shapeld` is a unique identifier assigned to each text annotation within the image editor. It serves as a reference to a specific annotation, enabling targeted deletion of the desired text element. By specifying the `shapeld` associated with the text annotation you want to remove, you can effectively delete it from the image editor.

To retrieve the inserted text annotations, you can utilize the `getShapeSetting` method, which provides a collection of annotations represented by `ShapeSettings`. This method allows you to access and work with the annotations that have been inserted into the image.

Here is an example of deleting a text in a button click using `deleteShape` method.

CSHTML

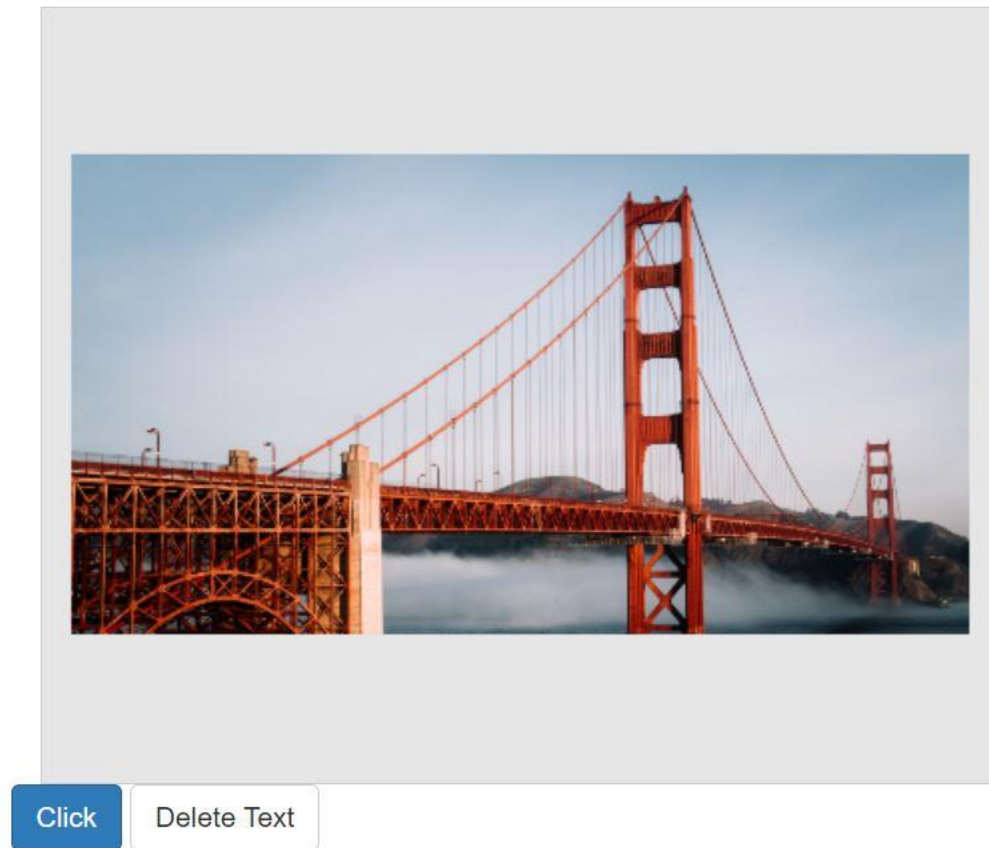
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
@Html.EJS().Button("delBtnClick").CssClass("e-primary").Content("Delete
Text").Click("deleteBtnClickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        var dimension = imageEditorObj.getImageDimension();
        imageEditorObj.drawText(dimension.x, dimension.y, 'Enter\nText');
    }
    function deleteBtnClickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.deleteShape('shape_1');
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
</style>
```

```
.e-img-editor-sample {  
    height: 80vh;  
    width: 100%;  
}  
@@media only screen and (max-width: 700px) {  
    .e-img-editor-sample {  
        height: 75vh;  
        width: 100%;  
    }  
}  
.control-wrapper {  
    height: 100%;  
}  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Customize font family and text color

The ShapeChanging event in the Image Editor control is triggered when a text annotation is being modified or changed through the toolbar interaction. This event provides an opportunity to make alterations to the text's color and font family by adjusting the relevant properties.

By leveraging the ShapeChanging event, you can enhance the customization options for text annotations and provide a more tailored and interactive experience within the Image Editor control.

Here is an example of changing the text's color and its font family using the ShapeChanging event.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").ShapeChanging("ShapeChanging").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function ShapeChanging(args) {
        if (args.currentShapeSettings.type === 'Text') {
            args.currentShapeSettings.color = 'red';
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        var dimension = imageEditorObj.getImageDimension();
        imageEditorObj.drawText(dimension.x, dimension.y, 'Enter\nText');
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
</style>
```

```

    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Output be like the below.

![ImageEditor Sample](images/image-editor-customize-font-color.jpg)

Add Additional font family

The FontFamily property in the Image Editor control provides the flexibility to incorporate supplementary font families, expanding your options for text styling and ensuring a broader range of fonts can be utilized within your design or content. The font value will be determined by the 'id' property.

By leveraging the FontFamily property, you can elevate the scope of customization for text annotations, enriching the user experience within the Image Editor control. This enhancement offers a more personalized and dynamic interaction, empowering users to tailor their text styles for a truly engaging editing experience.

Here is an example of adding additional font family to the text annotation using the FontFamily property.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").FontFamily(@ViewBag.fontFamily).Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }

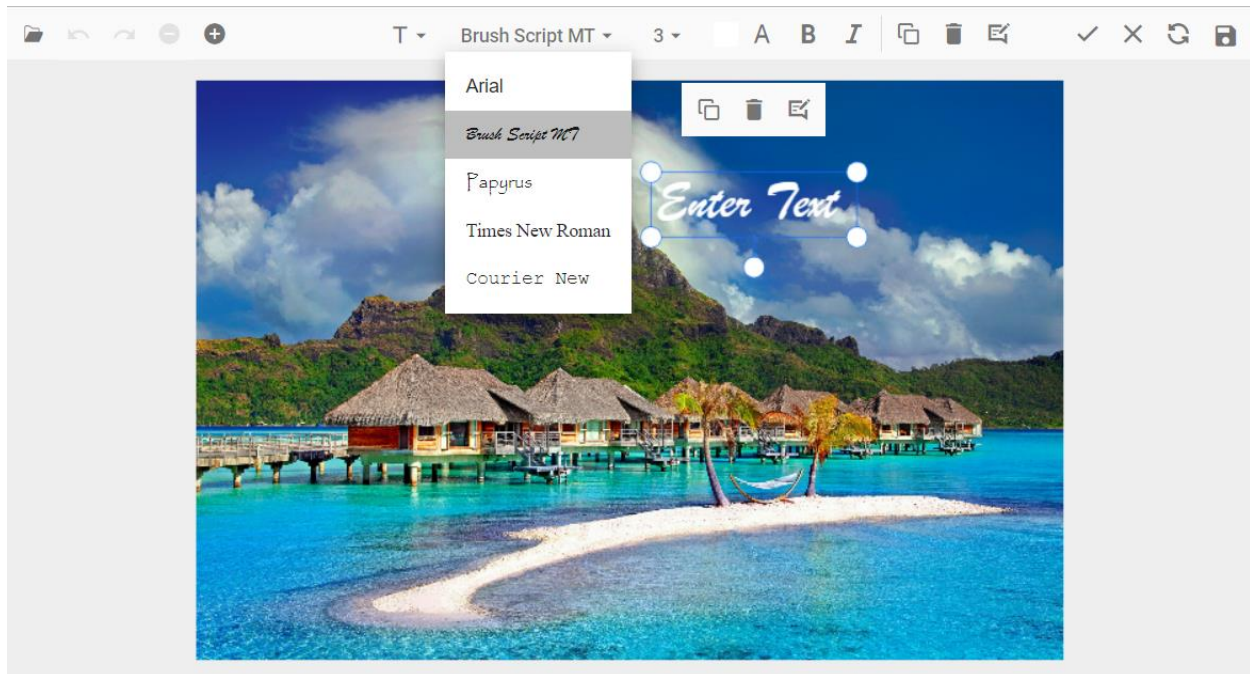
```

```
.e-img-editor-sample {  
    height: 80vh;  
    width: 100%;  
}  
@@media only screen and (max-width: 700px) {  
    .e-img-editor-sample {  
        height: 75vh;  
        width: 100%;  
    }  
}  
.control-wrapper {  
    height: 100%;  
}  
</style>
```

DEFAULT.CS

```
using Syncfusion.EJ2.ImageEditor;  
public ActionResult Default()  
{  
    ImageEditorFontFamily fontFamily = new ImageEditorFontFamily { Default =  
"Arial",  
        Items = new object[] {  
            new { id = "arial", text = "Arial" },  
            new { id = "brush script mt", text = "Brush Script MT" },  
            new { id = "papyrus", text = "Papyrus" },  
            new { id = "times new roman", text = "Times New Roman" },  
            new { id = "courier new", text = "Courier New" }  
        }  
    };  
    ViewBag.fontFamily = fontFamily;  
    return View();  
}
```

Output be like the below.



Freehand drawing

The Freehand Draw annotation tool in the Image Editor control is a versatile feature that allows users to draw and sketch directly on the image using mouse or touch input. This tool provides a flexible and creative way to add freehand drawings or annotations to the image.

The `freehandDraw` method is used to enable or disable the freehand drawing option in the Image Editor control.

Here is an example of using the `freeHandDraw` method in a button click event.

CSHTML

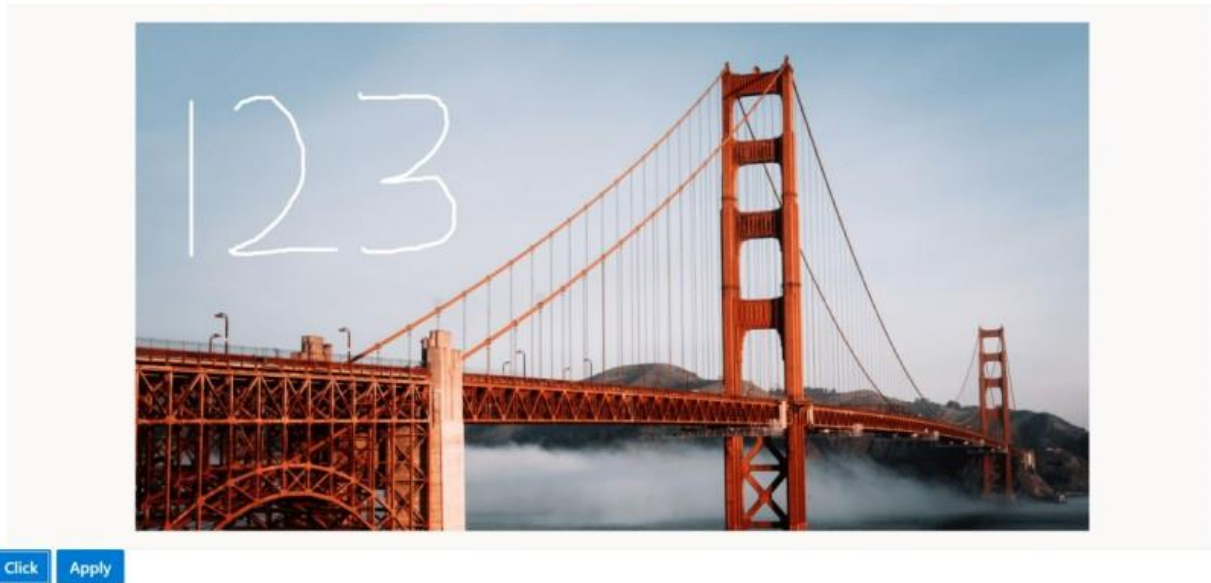
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
@Html.EJS().Button("applyBtnClick").CssClass("e-
primary").Content("Apply").Click("applyClickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
```

```
var imageEditorObj =  
ej.base.getComponent(document.getElementById('image-editor'), 'image-  
editor');  
    imageEditorObj.freeHandDraw(true);  
}  
function applyClickHandler() {  
    var imageEditorObj =  
ej.base.getComponent(document.getElementById('image-editor'), 'image-  
editor');  
    imageEditorObj.freeHandDraw(false);  
}  
</script>  
<style>  
    .image-editor {  
        margin: 0 auto;  
    }  
    .e-img-editor-sample {  
        height: 80vh;  
        width: 100%;  
    }  
    @@media only screen and (max-width: 700px) {  
        .e-img-editor-sample {  
            height: 75vh;  
            width: 100%;  
        }  
    }  
    .control-wrapper {  
        height: 100%;  
    }  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Adjust the stroke width and color

The `shapeChanging` event in the Image Editor control is triggered when a freehand annotation is being modified or changed through the toolbar interaction. This event provides an opportunity to make alterations to the freehand annotation's color and stroke width by adjusting the relevant properties.

By leveraging the `ShapeChanging` event, you can enhance the customization options for freehand annotations and provide a more tailored and interactive experience within the Image Editor control.

Here is an example of changing the freehand draw stroke width and color using the `ShapeChanging` event.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").ShapeChanging("ShapeChanging").Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function ShapeChanging(args) {
        if (args.currentShapeSettings.type === 'FreehandDraw') {
            args.currentShapeSettings.strokeColor = 'red';
        }
    }
</script>
<style>
```

```

.image-editor {
    margin: 0 auto;
}
.e-img-editor-sample {
    height: 80vh;
    width: 100%;
}
@@media only screen and (max-width: 700px) {
    .e-img-editor-sample {
        height: 75vh;
        width: 100%;
    }
}
.control-wrapper {
    height: 100%;
}
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Delete a freehand drawing

The `deleteShape` method in the Image Editor allows you to remove a freehand annotation from the image editor. To use this method, you need to pass the `shapeId` of the annotation as a parameter.

The `shapeId` is a unique identifier assigned to each freehand annotation within the image editor. It serves as a reference to a specific annotation, enabling targeted deletion of the desired annotation. By specifying the `shapeId` associated with the freehand annotation you want to remove, you can effectively delete it from the image editor.

To retrieve the inserted freehand annotations, you can utilize the `getShapeSetting` method, which provides a collection of annotations represented by `ShapeSettings`. This method allows you to access and work with the annotations that have been inserted into the image.

Here is an example of deleting a freehand annotation in a button click using `DeleteShape` method.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
@Html.EJS().Button("delBtnClick").CssClass("e-primary").Content("Delete
Click").Click("delclickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {

```

```

        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
    } else {
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
    }
}
function clickHandler() {
    var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
    imageEditorObj.freeHandDraw(true);
}
function delclickHandler() {
    var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
    imageEditorObj.deleteShape('pen_1');
}
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Shape annotation

The Image Editor control provides the ability to add shape annotations to an image. These shape annotations include rectangles, ellipses, arrows, paths, and lines, allowing you to highlight, emphasize, or mark specific areas or elements within the image.

Add a rectangle / ellipse / line / arrow / path

The `drawRectangle` method is used to insert a rectangle to the Image Editor control. Rectangle annotations are valuable tools for highlighting, emphasizing, or marking specific areas of an image to draw attention or provide additional context.

The `drawRectangle` method in the Image Editor control takes seven parameters to define the properties of the rectangle annotation:

- `x`: Specifies the x-coordinate of the top-left corner of the rectangle.
- `y`: Specifies the y-coordinate of the top-left corner of the rectangle.
- `width`: Specifies the width of the rectangle.
- `height`: Specifies the height of the rectangle.
- `strokeWidth`: Specifies the stroke width of the rectangle's border.
- `strokeColor`: Specifies the stroke color of the rectangle's border.
- `fillColor`: Specifies the fill color of the rectangle.

The `drawEllipse` method is used to insert an ellipse to the Image Editor control. Ellipse annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The `drawEllipse` method in the Image Editor control takes seven parameters to define the properties of the ellipse annotation:

- `x`: Specifies the x-coordinate of the center of the ellipse.
- `y`: Specifies the y-coordinate of the center of the ellipse.
- `radiusX`: Specifies the horizontal radius (radiusX) of the ellipse.
- `radiusY`: Specifies the vertical radius (radiusY) of the ellipse.
- `strokeWidth`: Specifies the width of the ellipse's stroke (border).
- `strokeColor`: Specifies the color of the ellipse's stroke (border).
- `fillColor`: Specifies the fill color of the ellipse.

The `drawLine` method is used to insert a line to the Image Editor control. Line annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The `drawLine` method in the Image Editor control takes seven parameters to define the properties of the ellipse annotation:

- `startX` - Specifies the x-coordinate of the start point.
- `startY` - Specifies the y-coordinate of the start point.
- `endX` - Specifies the x-coordinate of the end point.
- `endY` - Specifies the y-coordinate of the end point.
- `strokeWidth` - Specifies the stroke width of the line.
- `strokeColor` - Specifies the stroke color of the line.

The `drawArrow` method is used to insert an arrow to the Image Editor control. Arrow annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The `drawArrow` method in the Image Editor control takes three parameters to define the properties of the ellipse annotation:

- startX - Specifies the x-coordinate of the start point.
- startY - Specifies the y-coordinate of the start point.
- endX - Specifies the x-coordinate of the end point.
- endY - Specifies the y-coordinate of the end point.
- strokeWidth - Specifies the stroke width of the arrow.
- strokeColor - Specifies the stroke color of the arrow.
- arrowStart - Specifies the arrowhead as ImageEditorArrowHeadType at the start of arrow.
- arrowEnd - Specifies the arrowhead as ImageEditorArrowHeadType at the end of the arrow.

The `drawPath` method is used to insert a path to the Image Editor control. Path annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The `drawPath` method in the Image Editor control takes three parameters to define the properties of the ellipse annotation:

- points - Specifies collection of x and y coordinates as ImageEditorPoint to draw a path.
- strokeWidth - Specifies the stroke width of the path.
- strokeColor - Specifies the stroke color of the path.

Here is an example of inserting rectangle, ellipse, arrow, path, and line in a button click event.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        var dimension = imageEditorObj.getImageDimension();
        imgObj.drawRectangle(dimension.x, dimension.y);
        imgObj.drawEllipse(dimension.x, dimension.y);
        imgObj.drawLine(dimension.x, dimension.y);
        imgObj.drawArrow(dimension.x, dimension.y + 10, dimension.x + 50,
dimension.y + 10, 10);
        imgObj.drawPath([ { x: dimension.x, y: dimension.y }, { x:
dimension.x + 50, y: dimension.y + 50 }, { x: dimension.x + 20, y:
dimension.y + 50 } ], 8);
    }
</script>
```

```

    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Delete a shape

The `deleteShape` method in the Image Editor allows you to remove a shape annotation from the image editor. To use this method, you need to pass the `shapeId` of the annotation as a parameter.

The `shapeId` is a unique identifier assigned to each shape annotation within the image editor. It serves as a reference to a specific annotation, enabling targeted deletion of the desired annotation. By specifying the `shapeId` associated with the shape annotation you want to remove, you can effectively delete it from the image editor.

To retrieve the inserted shape annotations, you can utilize the `getShapeSetting` method, which provides a collection of annotations represented by `ShapeSettings`. This method allows you to access and work with the annotations that have been inserted into the image.

Here is an example of deleting rectangle, ellipse, arrow, path, and line in a button click event.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
@Html.EJS().Button("delBtnClick").CssClass("e-primary").Content("Delete
Text").Click("deleteBtnClickHandler").Render()
<script>
    function created() {

```

```

        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
        function clickHandler() {
            var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
            var dimension = imageEditorObj.getImageDimension();
            imageEditorObj.drawRectangle(dimension.x, dimension.y);
        }
        function deleteBtnClickHandler() {
            var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
            imageEditorObj.deleteShape('shape_1');
        }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Image annotation

The image annotation feature in the Image Editor provides the capability to add and customize images directly onto the image. With this feature, you can easily insert image or icons at specific locations

within the image and customize various aspects of the image to meet your requirements. You have control over the customization options including rotate, flip, transparency for the image annotation.

Add an image annotation

The `drawImage` method serves the purpose of inserting an image into the Image Editor control, allowing for image annotations to be added. These image annotations can be used for various purposes, such as adding logos, watermarks, or decorative elements to the image.

The `drawImage` method in the Image Editor control takes six parameters to define the properties of the image annotation:

- `data`: Specified the image data or url of the image to be inserted.
- `x`: Specifies the x-coordinate of the top-left corner of the image.
- `y`: Specifies the y-coordinate of the top-left corner of the image.
- `width`: Specifies the width of the image.
- `height`: Specifies the height of the image.
- `isAspectRatio`: Specifies whether the image is rendered with aspect ratio or not.

In the following example, you can use the `drawImage` method in the button click event.

CSHTML

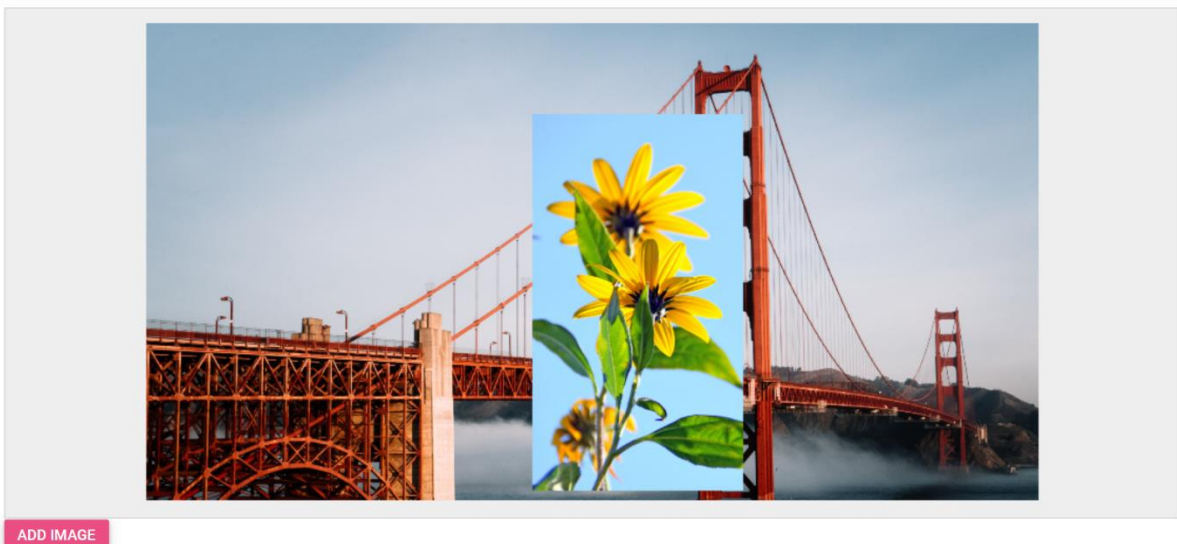
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-primary").Content("Add
Image").Click("btnClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function btnClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.drawImage('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png', 500, 100, 200, 80, true, 0);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
</style>
```

```
}
@@media only screen and (max-width: 700px) {
    .e-img-editor-sample {
        height: 75vh;
        width: 100%;
    }
}
.control-wrapper {
    height: 100%;
}
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Transform in the ASP.NET MVC Image Editor control

The Image Editor provides a range of transformation options for manipulating both the image and its annotations. These options include rotation, flipping, zooming, and panning. These transformations offer flexibility in adjusting the image and enhancing its visual appearance.

Rotate an image

The Image Editor allows to rotate the image and its annotations by a specific number of degrees clockwise or anti-clockwise using `rotate` method. This method takes a single parameter: the angle of rotation in degrees. A positive value will rotate the image clockwise, while a negative value will rotate it anti-clockwise.

Here is an example of rotating an image in a button click event.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.rotate(90);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

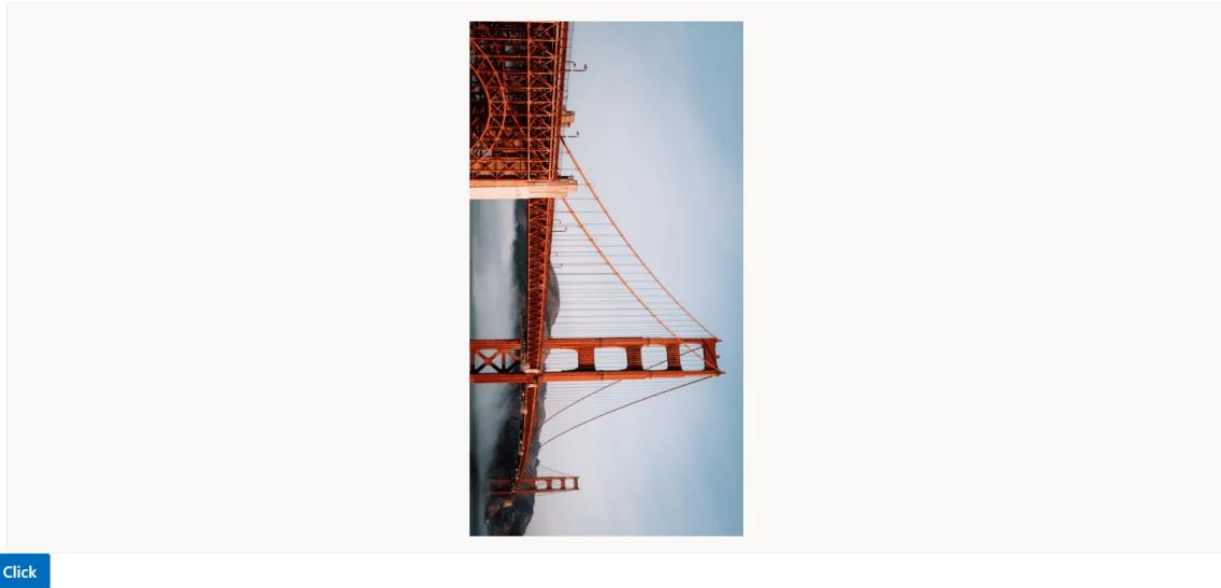
DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Output be like the below.



Flip an image

The Image Editor provides the `flip` method, which allows you to flip both the image and its annotations either horizontally or vertically. This method takes a single parameter of type `ImageEditorDirection`, which specifies the direction in which the flip operation should be applied.

The `Direction` parameter accepts two values: 'Horizontal' and 'Vertical'. When you choose 'Horizontal', the image and annotations will be flipped along the horizontal axis, resulting in a mirror effect. On the other hand, selecting 'Vertical' will flip them along the vertical axis, producing a vertical mirror effect.

Here is an example of flipping an image in a button click event.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("btnClick").CssClass("e-
primary").Content("Click").Click("clickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
```

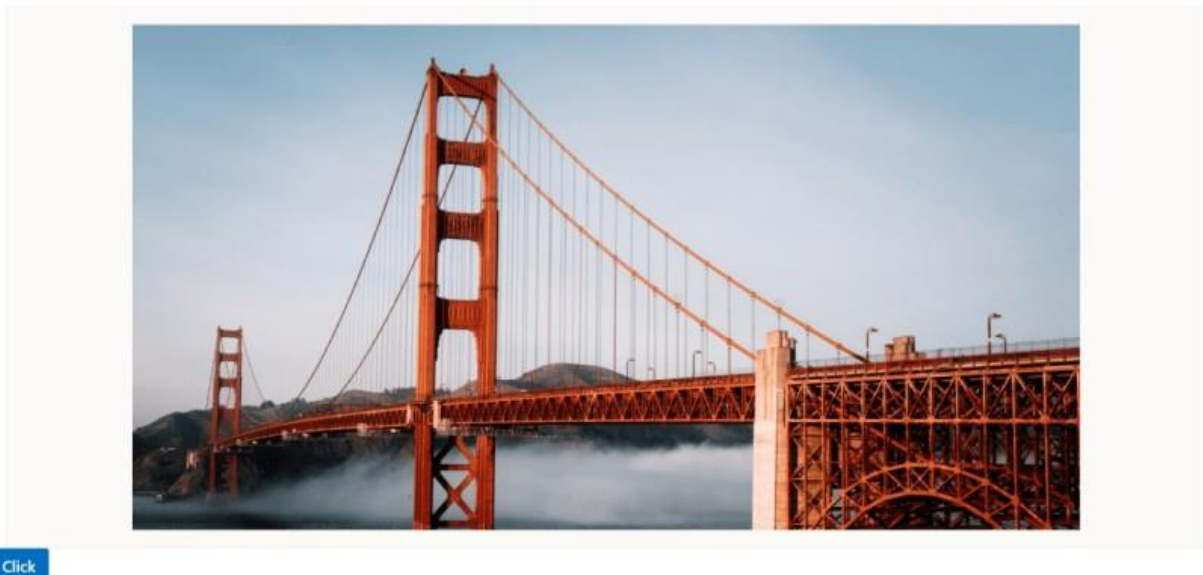


```
        imageEditorObj.flip("Horizontal"); // Horizontal flip
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Straighten an image

The straightening feature in an Image Editor allows users to adjust an image by rotating it clockwise or counter clockwise. The rotating degree value should be within the range of -45 to +45 degrees for accurate straightening. Positive values indicate clockwise rotation, while negative values indicate

counter clockwise rotation. The Image Editor control includes a `straightenImage` method, which allows you to adjust the degree of an image. This method takes one parameter that define how the straightening should be carried out:

- **degree:** Specifies the amount of rotation for straightening the image. Positive values indicate clockwise rotation, while negative values indicate counterclockwise rotation.

Here is an example of straightening the image.

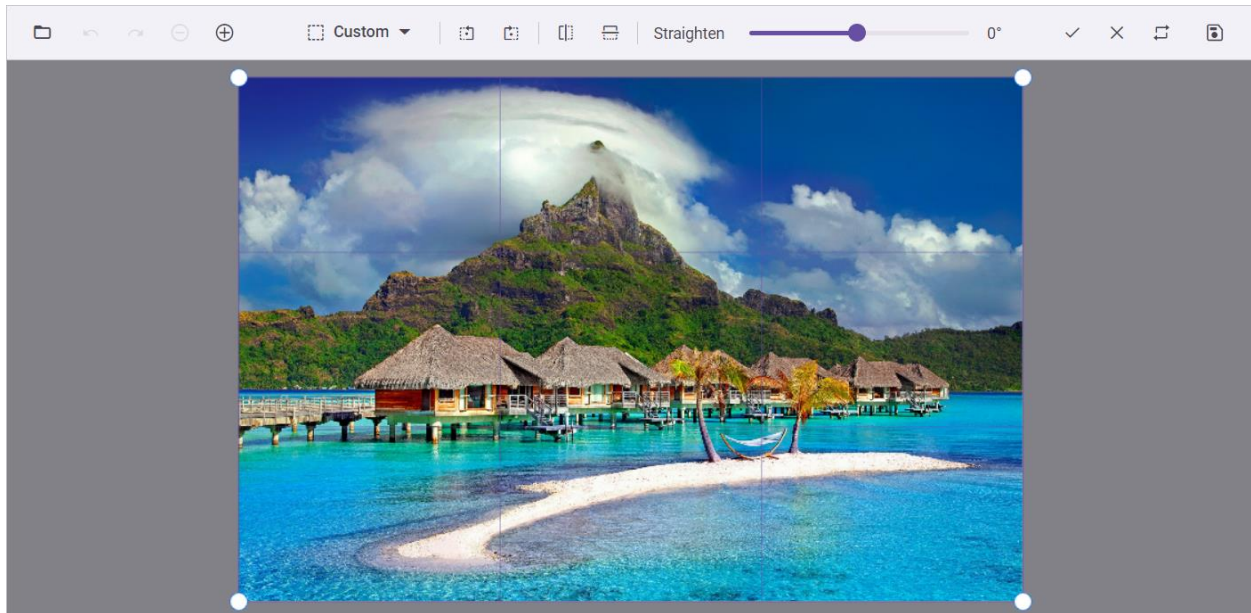
CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Zoom in or out an image

The Image Editor allows to magnify an image using the **zoom** method. This method allows one to zoom in and out of the image and provides a more detailed view of the image's hidden areas. This method takes two parameters to perform zooming.

- **zoomFactor** - Specifies a value to controlling the level of magnification applied to the image.
- **zoomPoint** - Specifies x and y coordinates of a point as **ImageEditorPoint** on image to perform zooming.

Here is an example of zooming an image in a button click event.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("zoomIn").CssClass("e-primary").Content("Zoom
In").Click("zoomInClick").Render()
@Html.EJS().Button("zoomOut").CssClass("e-primary").Content("Zoom
Out").Click("zoomOutClickHandler").Render()
<script>
    var zoomSettings = { maxZoomFactor: 30, minZoomFactor: 0.1 };
    var zoomLevel = 1;
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
```

```

        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
    }
}
function zoomInClick() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    if (this.zoomLevel < 1) {
        this.zoomLevel += 0.1;
    } else {
        this.zoomLevel += 1;
    }
    value = this.zoomSettings.maxZoomFactor;
    if (this.zoomLevel > value) {
        this.zoomLevel = value;
    }
    imgObj.zoom(this.zoomLevel); // Zoom in
}
function zoomOutClickHandler() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    if (this.zoomLevel <= 1) {
        this.zoomLevel -= 0.1;
    } else {
        this.zoomLevel -= 1;
    }
    value = this.zoomSettings.minZoomFactor;
    if (this.zoomLevel < value) {
        this.zoomLevel = value;
    }
    imgObj.zoom(this.zoomLevel); // Zoom out
}
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

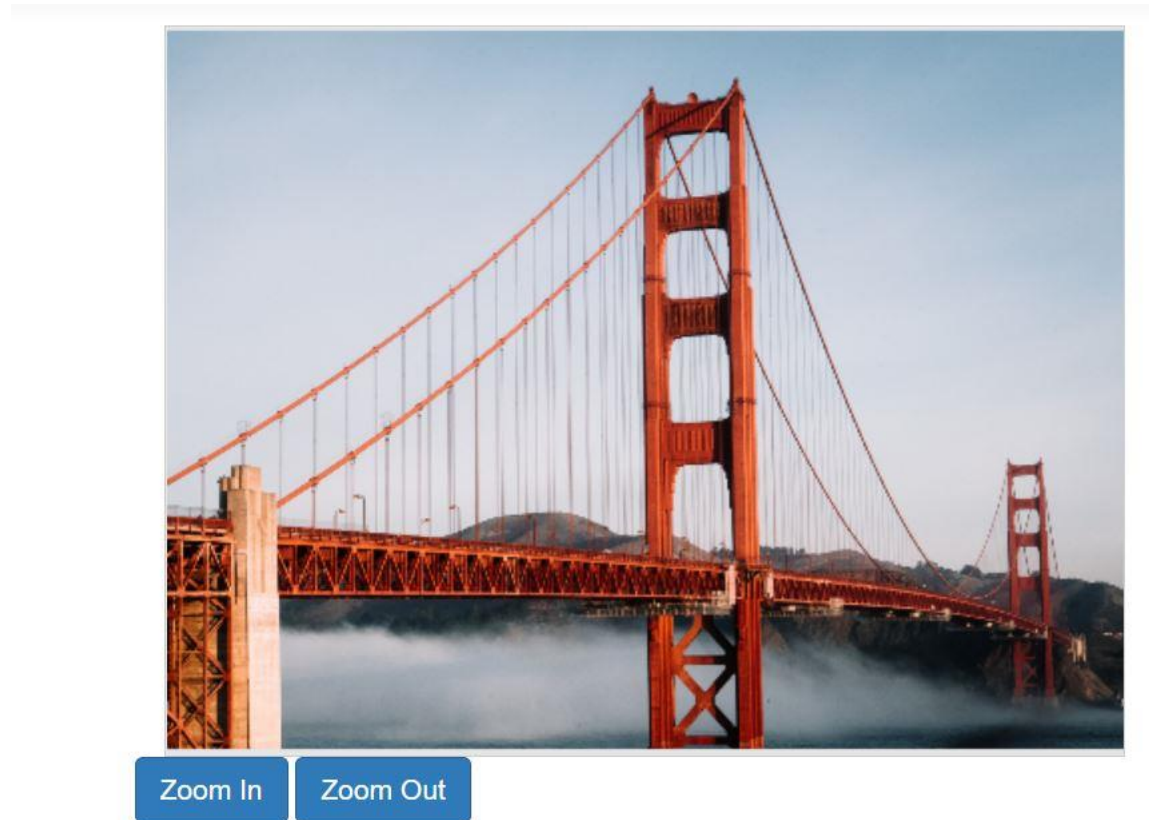
```

public ActionResult Default()
{

```

```
return View();  
}
```

Output be like the below.



Maximum and Minimum zoom level

The `maxZoomFactor` property is a useful feature in the Image Editor that allows you to define the maximum level of zoom permitted for an image. This property sets a limit on how much the image can be magnified, preventing excessive zooming that may result in a loss of image quality or visibility.

By default, the `minZoomFactor` value is set to 10, meaning that the image can be zoomed in up to 10 times its original size. This ensures that the zooming functionality remains within reasonable bounds and maintains the integrity of the image.

The `maxZoomFactor` property allows you to specify the minimum level of zoom that is allowed for an image. By setting this property, you can prevent the image from being zoomed out beyond a certain point, ensuring that it remains visible and usable even at the smallest zoom level.

By default, the `maxZoomFactor` value is set to 0.1, meaning that the image can be zoomed out up to 10 times its original size.

Here is an example of specifying `minZoomFactor` and `maxZoomFactor` property in `zoomSettings` options in an image editor.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
```

```

    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("zoomIn").CssClass("e-primary").Content("Zoom
In").Click("zoomInClick").Render()
@Html.EJS().Button("zoomOut").CssClass("e-primary").Content("Zoom
Out").Click("zoomOutClickHandler").Render()
<script>
    var zoomSettings = { maxZoomFactor: 30, minZoomFactor: 0.1 };
    var zoomLevel = 1;
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function zoomInClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        if (this.zoomLevel < 1) {
            this.zoomLevel += 0.1;
        } else {
            this.zoomLevel += 1;
        }
        value = this.zoomSettings.maxZoomFactor;
        if (this.zoomLevel > value) {
            this.zoomLevel = value;
        }
        imgObj.zoom(this.zoomLevel); // Zoom in
    }
    function zoomOutClickHandler() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        if (this.zoomLevel <= 1) {
            this.zoomLevel -= 0.1;
        } else {
            this.zoomLevel -= 1;
        }
        value = this.zoomSettings.minZoomFactor;
        if (this.zoomLevel < value) {
            this.zoomLevel = value;
        }
        imgObj.zoom(this.zoomLevel); // Zoom out
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
    }

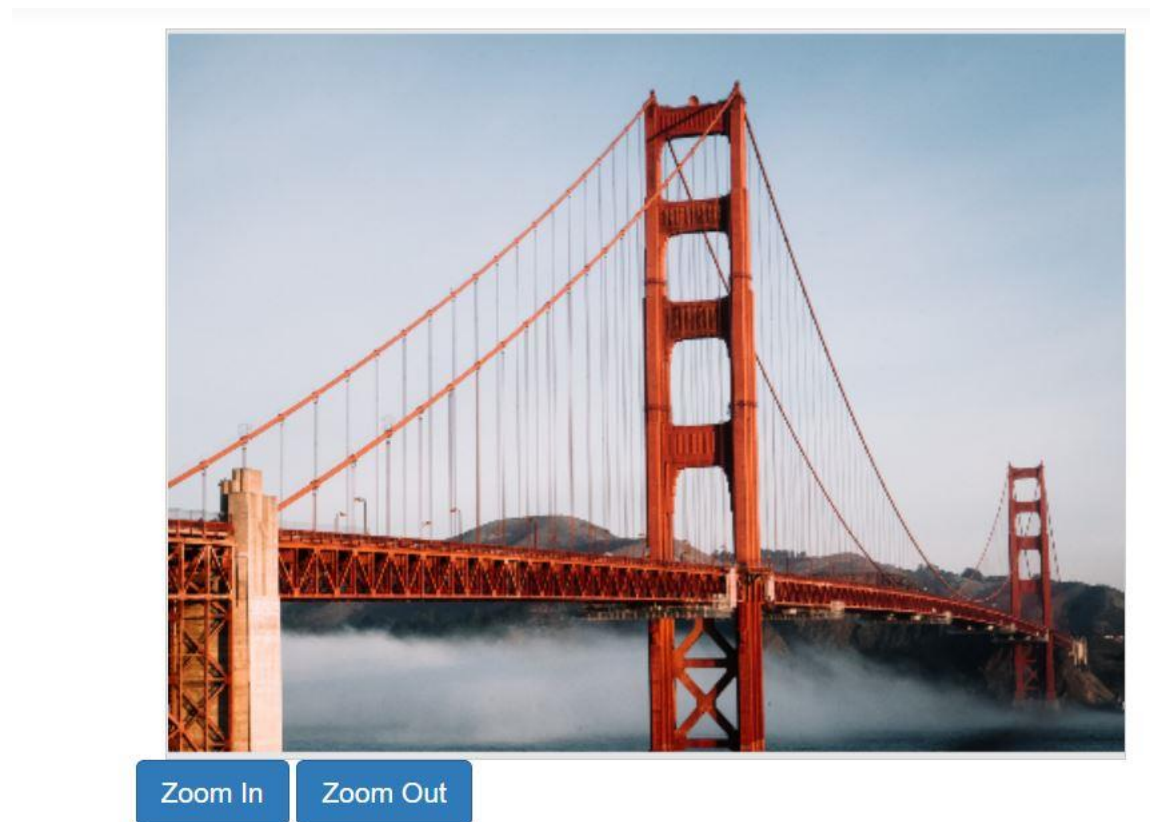
```

```
width: 100%;  
}  
@@media only screen and (max-width: 700px) {  
  .e-img-editor-sample {  
    height: 75vh;  
    width: 100%;  
  }  
}  
.control-wrapper {  
  height: 100%;  
}  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Panning an image

The Image Editor allows to pan an image when the image exceeds the canvas size or selection range. When zooming in on an image or applying a selection for cropping, it is common for the image to exceed the size of the canvas or exceed the selection range. So, the panning is used to view the entire image, by clicking on the canvas and dragging it in the direction they want to move.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("zoomIn").CssClass("e-primary").Content("Zoom
In").Click("zoomInClickHandler").Render()
@Html.EJS().Button("zoomOut").CssClass("e-primary").Content("Zoom
Out").Click("zoomOutClickHandler").Render()
@Html.EJS().Button("pan").CssClass("e-
primary").Content("Pan").Click("panClickHandler").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function zoomInClickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.zoom(.1); // Zoom in
        imageEditorObj.pan(true);
    }
    function zoomOutClickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.zoom(-.1); // Zoom out
    }
    function panClickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.zoom(.1); // Zoom in
        imageEditorObj.pan(true);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;

```



```
        width: 100%;  
    }  
}  
.control-wrapper {  
    height: 100%;  
}  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Panning event

The **panning** event is activated when the user begins dragging the image within the canvas. This event provides an opportunity to perform specific actions, like adjusting the position of an image, in response to the gesture of panning. And this event uses **panEventArgs** to handle the panning action when the user starts dragging the image.

The parameter available in the **panEventArgs** events are,

- **PanEventArgs.startPoint** - The x and y coordinates as **ImageEditorPoint** for the start point.
- **PanEventArgs.endpoint** - The x and y coordinates as **ImageEditorPoint** for the end point.
- **PanEventArgs.cancel** - Specifies the boolean value to cancel the panning action.

Zooming event

The **zooming** event is triggered when performing zooming the image. This event can be used to perform certain actions, such as updating the position of the image. This event is passed an object that contains information about the zooming event, such as the amount of zooming performed. And this event uses **ZoomEventArgs** to handle the zooming action in the image.

The parameter available in the Zooming event is,

- **ZoomEventArgs.zoomPoint** - The x and y coordinates as **ImageEditorPoint** for the zoom point.
- **ZoomEventArgs.previousZoomFactor** - The previous zoom factor applied in the image editor.
- **ZoomEventArgs.currentZoomFactor** - The current zoom factor to be applied in the image editor.
- **ZoomEventArgs.cancel** - Specify a boolean value to cancel the zooming action.
- **ZoomEventArgs.zoomTrigger** - The type of zooming performed in the image editor.

Rotating event

The **rotating** event is triggered when performing rotating the image. This event is passed an object that contains information about the rotating event, such as the amount of rotation performed. And this event uses **RotateEventArgs** to handle the rotating action in the image.

The parameter available in the Rotating event is,

- **RotateEventArgs.previousDegree**: The degree of rotation before the recent rotation action was applied in the Image Editor.

- `RotateEventArgs.currentDegree`: The current degree of rotation after the rotation action has been performed in the Image Editor.

`RotateEventArgs.cancel` – Specifies a boolean value to cancel the rotating action.

Flipping event

The `flipping` event is triggered when performing flipping the image. This event is passed an object that contains information about the flipping event, such as the amount of flip performed. And this event uses `FlipEventArgs` to handle the flipping action in the image.

The parameter available in the `flipping` event is,

- `FlipEventArgs.direction` - The flip direction as `ImageEditorDirection` to be applied in the image editor.
- `FlipEventArgs.cancel` - Specifies a boolean value to cancel the flip action.

Toolbar in the ASP.NET MVC Image Editor control

The toolbars in the Image Editor are a key component for interacting with and editing images. They provide a range of tools and options that can be customized to suit the needs and preferences. Add or remove items from the toolbar to create a personalized set of tools, or they can even create their own custom toolbar from scratch. This flexibility and customization allow them to create a unique image editing experience that is tailored to their specific needs and workflow.

In the Image Editor, the toolbar property provides the ability to customize the toolbar by adding or removing items, as well as defining a completely custom toolbar. This feature is valuable for creating a personalized image editing experience that aligns with specific requirements and workflows.

Built-in toolbar items

Specifies the toolbar items to perform UI interactions. Refer to the built-in toolbar items for the default value.

- Crop
- Transform
- Annotate
- ZoomIn
- ZoomOut
- Open
- Reset
- Save
- Pan

Add a custom toolbar items

The `toolbar` property in the Image Editor allows to add or remove toolbar items to include only the tools they frequently use, streamlining the editing process and reducing clutter.

Here is an example of adding custom toolbar items to rotate and flip transformation using `toolbar` property.

CSHTML

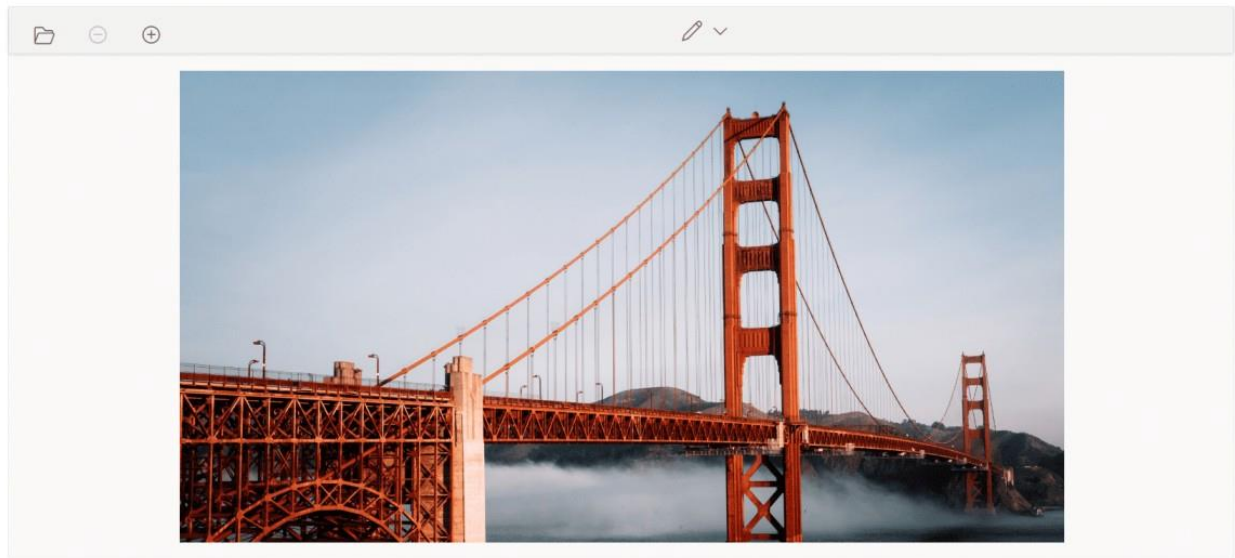
```
<div class="col-lg-12 control-section e-img-editor-sample">
```

```
@Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { "Annotate", "Line", "Rectangle", "Text", "ZoomIn", "ZoomOut"
}).Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Show or hide a toolbar

The **toolbar** property controls the visibility of the toolbar in the Image Editor. When the toolbar property is set to an empty list, the toolbar is hidden. Conversely, if the toolbar property contains a list of items, the toolbar is shown, displaying the specified items. This feature provides flexibility for users to personalize their image editing experience.

Here is an example of hiding the toolbar of the image editor using **toolbar** property.

CSHTML

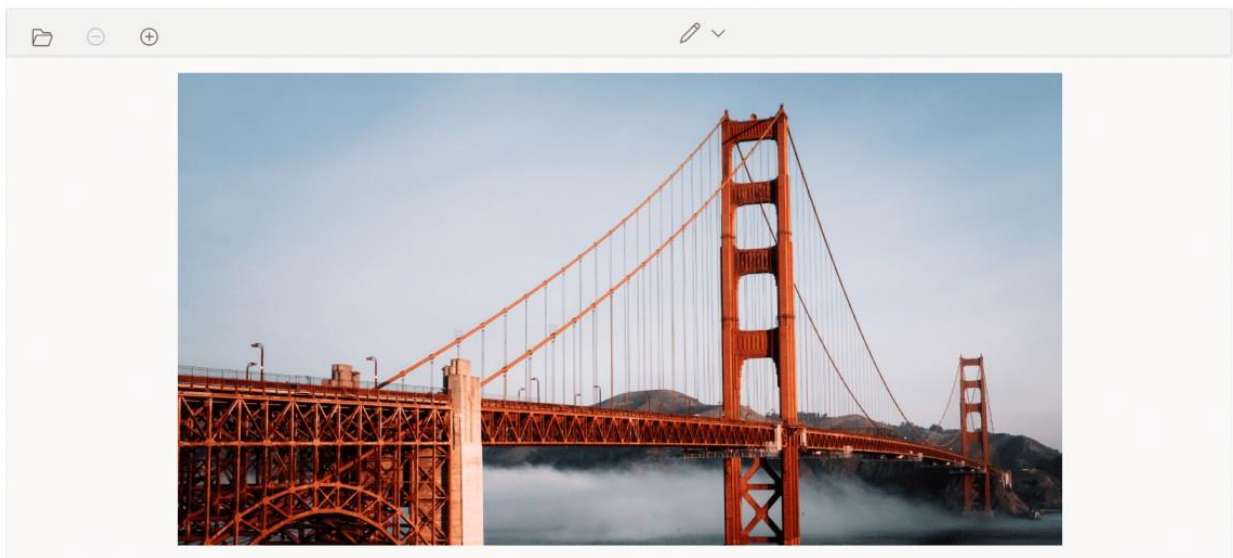
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { "Annotate", "Line", "Rectangle", "Text", "ZoomIn", "ZoomOut"
}).Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
</style>
```

```
@@media only screen and (max-width: 700px) {  
    .e-img-editor-sample {  
        height: 75vh;  
        width: 100%;  
    }  
}  
.control-wrapper {  
    height: 100%;  
}
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Show or hide a toolbar item

The **toolbar** property is utilized to control the visibility of toolbar items in the Image Editor. By default, the toolbar property includes the default toolbar items. So, if you wish to hide the default toolbar items then you need to explicitly define the required items using **toolbar** property. This allows you to customize the toolbar by displaying only the specific items you require, tailoring the editing experience to your preferences.

Here is an example of hiding the cropping and selection toolbar items using **toolbar** property.

CSHTML

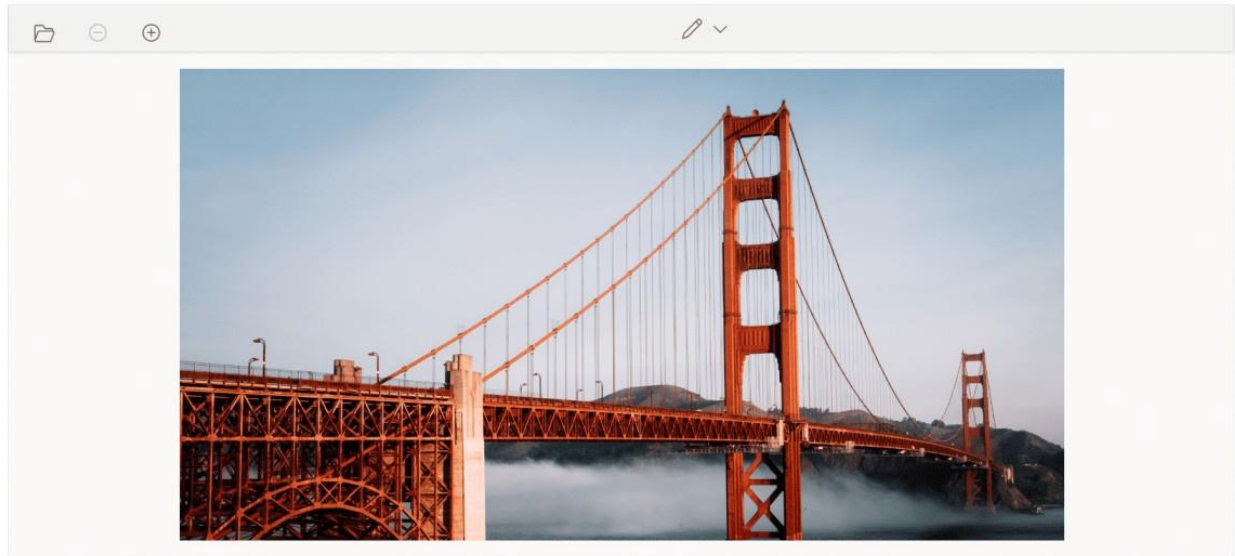
```
<div class="col-lg-12 control-section e-img-editor-sample">  
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new  
        string[] { "Annotate", "Line", "Rectangle", "Text", "ZoomIn", "ZoomOut"  
    }).Render()  
</div>
```

```
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



Enable or disable a toolbar item

The `toolbar` property is employed to enable or disable toolbar items in the Image Editor. By default, the `toolbar` property includes the default toolbar items, and these items cannot be disabled. However, if you have defined custom toolbar items using the `toolbarItemModel`, you can enable or disable them by configuring their respective properties within the `toolbar` property. This provides the flexibility to control the availability and functionality of custom toolbar items based on your specific requirements.

Here is an example of disabling the custom toolbar item using `toolbar` property.

Enable or disable a contextual toolbar item

The `toolbarItems` property in the `toolbarEventArgs` is used to enable or disable contextual toolbar items in the Image Editor. To enable or disable the default toolbar items, you can accomplish this by setting the `Disabled` property to `true` in the `ImageEditorToolbarItemModel` within the `ToolbarItems` property. This allows you to selectively enable or disable specific default toolbar items based on your requirements, providing a customized toolbar experience in the Image Editor.

Toolbar created event

The `toolbarCreated` event is triggered after the toolbar is created in the Image Editor. This event can be useful when you need to perform any actions or make modifications to the toolbar once it is fully initialized and ready for interaction. By subscribing to the `toolbarCreated` event, you can access the toolbar object and perform tasks such as adding event handlers, customizing the appearance, or configuring additional functionality.

Toolbar item clicked event

The `toolbarItemClicked` event is triggered when a toolbar item is clicked in the Image Editor. This event is particularly useful when you have added custom options to both the main toolbar and contextual toolbar, as it allows you to capture the user's interaction with those custom options. By subscribing to the `toolbarItemClicked` event, you can execute specific actions or handle logic based on the toolbar item that was clicked.

Here is an example of toolbar item clicking event using `toolbarItemClicked` property.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").ToolBarItemClicked("toolbarItemClicked").ToolBar(
new string[] { "Annotate", "Line", "Rectangle", "Text", "ZoomIn", "ZoomOut"
}).Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function toolbarItemClicked (args) {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (args.item.text === 'Text') {
            imageEditorObj.rotate(90);
        }
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```


Toolbar template

The `toolbarTemplate` property in the Image Editor provides the capability to fully customize the toolbar by supplying a custom template. This feature is valuable when you want to create a distinct and personalized image editing experience that goes beyond the default toolbar or the customizable toolbar options offered by the Image Editor. By defining a custom template for the toolbar, you have complete control over its layout, appearance, and functionality. This empowers you to design a unique and tailored toolbar that aligns perfectly with your specific requirements and desired user experience.

Here is an example of using `toolbarTemplate` to render only the button to toggle the freehand draw option.

The toolbar of the Image Editor can be replaced with the user specific UI using the `toolbarTemplate` property.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").ToolbarTemplate("#toolTemplate").Render()
</div>
<script id="toolTemplate" type="text/x-template">
    <button id='custombtn' class="e-btn"
onclick="clickHandler()">Custom</button>
</script>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function clickHandler() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        imageEditorObj.freeHandDraw(true);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
</style>
```

```

    }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

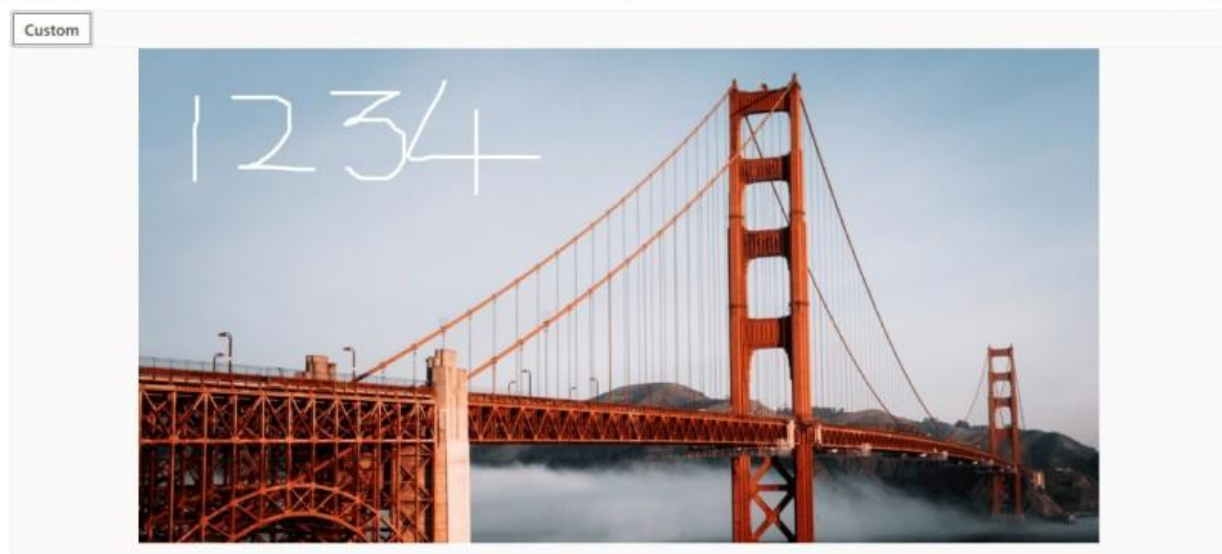
DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Output be like the below.



Customize contextual toolbar

The `toolbarUpdating` event is triggered when inserting or selecting annotations, which opens the contextual toolbar in the Image Editor. Within this event, the `toolbarItems` property in the `ToolbarEventArgs` is utilized to add or remove contextual toolbar items.

In the following example, the contextual toolbar for rectangle will be rendered with only stroke color by excluding fill color and stroke width using `toolbarUpdating` event.

CSHTML

```

<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").ToolbarUpdating("clickHandler").Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {

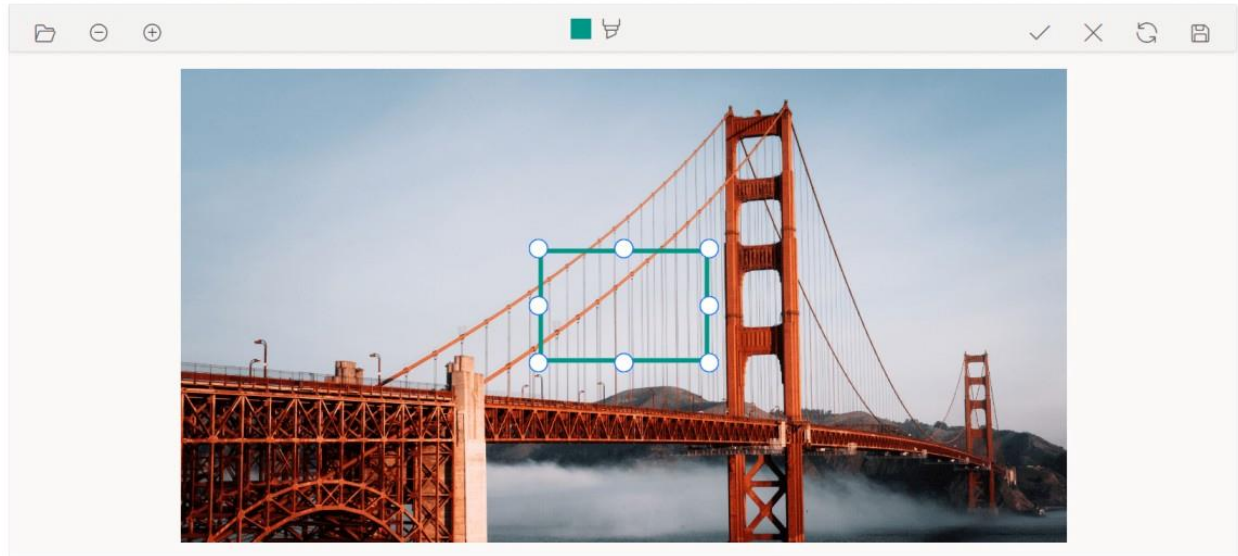
```

```
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-  
editor/images/flower.png');  
    } else {  
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-  
editor/images/bridge.png');  
    }  
}  
function clickHandler(args) {  
    if (args.toolbarType === 'shapes') {  
        args.toolbarItems = ['strokeColor'];  
    }  
}  
</script>  
<style>  
    .image-editor {  
        margin: 0 auto;  
    }  
    .e-img-editor-sample {  
        height: 80vh;  
        width: 100%;  
    }  
    @@media only screen and (max-width: 700px) {  
        .e-img-editor-sample {  
            height: 75vh;  
            width: 100%;  
        }  
    }  
    .control-wrapper {  
        height: 100%;  
    }  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Quick access toolbar in the ASP.NET MVC Image Editor control

The quick access toolbars in the Image Editor play a vital role in facilitating interactions with annotations like Rectangle, Ellipse, Line, Arrow, and Path. These toolbars offer a diverse array of tools and options that can be tailored to match the specific requirements and preferences associated with each annotation type. The toolbar is only displayed when an annotation is selected, ensuring a focused and contextual user experience. Users have the flexibility to add or remove items from the toolbar, allowing them to create a personalized set of tools. Additionally, users can also build a completely custom toolbar from the ground up, providing them with complete control over the available options and functionality.

Add a custom toolbar item

The quick access toolbar that appears when inserting annotations in the Image Editor is customizable using the `QuickAccessToolbarOpening` event. This event is triggered when the quick access toolbar is opened, allowing you to modify its contents. To add additional toolbar items to the quick access toolbar, you can access the `ToolbarItems` property of the `QuickAccessToolbarEventArgs` object within the event handler. By adding or removing items from the `ToolbarItems` property based on the `Item` property, you can customize the options available in the quick access toolbar according to your needs. This gives you the ability to extend the functionality of the quick access toolbar and provide additional tools and options for working with inserted annotations.

Here is an example of adding the custom toolbar item to the quick access toolbar.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-
editor").Created("created").QuickAccessToolbarOpen("quickAccessToolbarOpen")
.Render()
</div>
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        dh(ej.dbaseBrowser.isDevice) {
```

```
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-  
editor/images/flower.png');  
    } else {  
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-  
editor/images/bridge.png');  
    }  
}  
function quickAccessToolbarOpen(args) {  
    args.toolbarItems = ['Clone'];  
}  
</script>  
<style>  
    .image-editor {  
        margin: 0 auto;  
    }  
    .e-img-editor-sample {  
        height: 80vh;  
        width: 100%;  
    }  
    @@media only screen and (max-width: 700px) {  
        .e-img-editor-sample {  
            height: 75vh;  
            width: 100%;  
        }  
    }  
    .control-wrapper {  
        height: 100%;  
    }  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Undo Redo in the ASP.NET MVC Image Editor control

The undo and redo functionalities provide a way to reverse and repeat editing actions performed on an image. These features are essential for maintaining control and flexibility during the editing process.

In an image editor, the undo and redo history typically have a limited capacity, and the number of steps that can be stored is 16 steps, meaning that the editor keeps track of the most recent 16 actions performed on the image. Once the history reaches its maximum capacity, any new actions beyond the 16th step will result in the removal of the oldest action from the history.

Undo the action

The undo action in an image editor allows users to revert the most recent editing action or a series of actions back to their previous state. When the undo command is triggered, the image editor undoes the last applied modification, effectively restoring the image to its state before the action was performed. The undo action is useful for correcting mistakes, removing unwanted changes, or exploring different editing options without permanently altering the image.

Redo the Action

The redo action in an image editor allows users to reapply previously undone actions or modifications to the image. When the redo command is triggered, the image editor reapplies the last action that was undone, bringing the image back to the state it was in after the action was initially applied. The redo is useful when users want to repeat an action that was previously undone or restore changes that were temporarily reversed.

Here is an example of undoing and redoing the action using the `undo` and `redo` method.

CSHTML

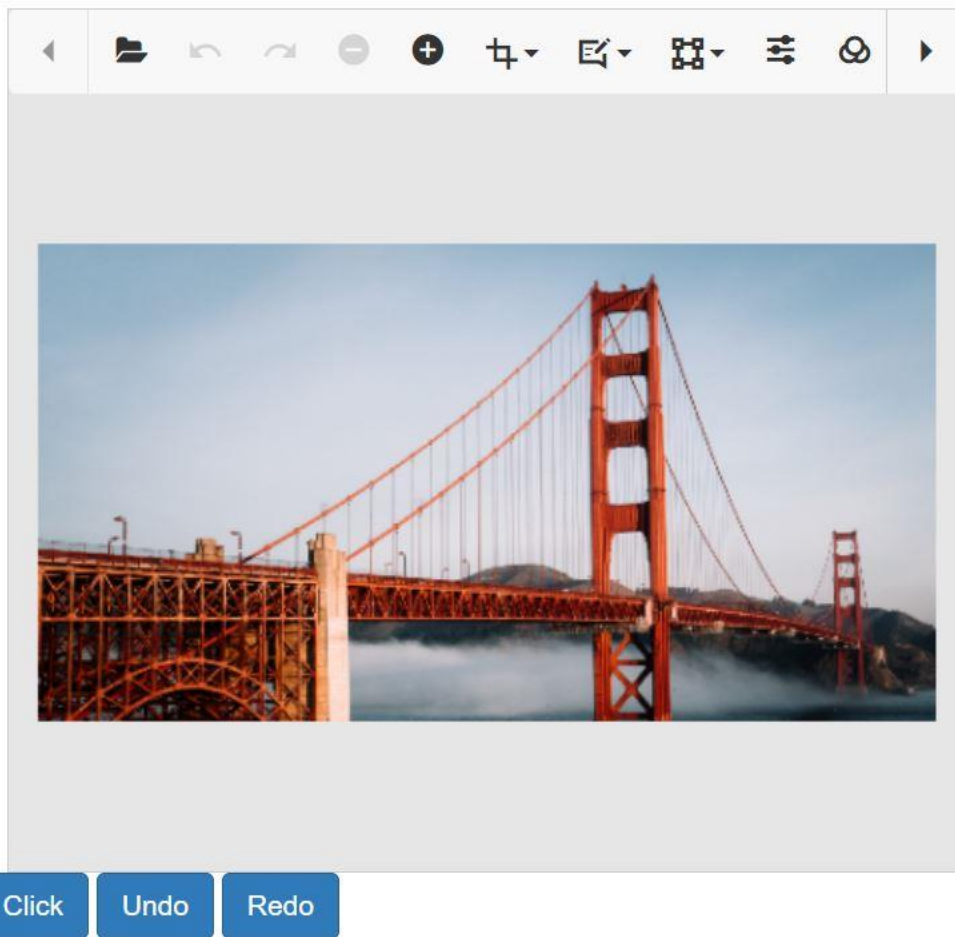
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Render()
</div>
@Html.EJS().Button("undoClick").CssClass("e-
primary").Content("Undo").Click("undoClick").Render()
@Html.EJS().Button("redoClick").CssClass("e-
primary").Content("Redo").Click("redoClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function undoClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.undo();
    }
    function redoClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.redo();
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
</style>
```

```
.control-wrapper {  
    height: 100%;  
}  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Filters in the ASP.NET MVC Image Editor control

Filters are pre-defined effects that can be applied to an image to alter its appearance or mood. Image filters can be used to add visual interest or to enhance certain features of the image. Some common types of image filters include cold, warm, chrome, sepia, and invert. This can be done by either using the toolbar or the `applyImageFilter` method which takes a single parameter: the filter applied to an image.

Apply filter effect

The `applyImageFilter` method is utilized to apply filters to an image. By passing the desired filter type as the first parameter of the method, specified as `ImageFilterOption` the method applies the corresponding filter to the image. This allows for easy and convenient application of various filters to enhance or modify the image based on the chosen filter type.

The `applyImageFilter` method is used to perform filtering by specifying the type of filter as `ImageFilterOption` and send it a first parameter of the method.

Here is an example of filtering using the `applyImageFilter` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("chromebtnClick").CssClass("e-
primary").Content("Chrome").Click("chromeClick").Render()
@Html.EJS().Button("coldbtnClick").CssClass("e-
primary").Content("Cold").Click("coldClick").Render()
@Html.EJS().Button("warmbtnClick").CssClass("e-
primary").Content("Warm").Click("warmClick").Render()
@Html.EJS().Button("grayScalebtnClick").CssClass("e-
primary").Content("GrayScale").Click("grayScaleClick").Render()
@Html.EJS().Button("sepiabtnClick").CssClass("e-
primary").Content("Sepia").Click("sepiaClick").Render()
@Html.EJS().Button("invertbtnClick").CssClass("e-
primary").Content("Invert").Click("invertClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function chromeClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.applyImageFilter(ej.imageeditor.ImageFilterOption.Chrome);
    }
    function coldClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.applyImageFilter(ej.imageeditor.ImageFilterOption.Cold);
    }
    function warmClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.applyImageFilter(ej.imageeditor.ImageFilterOption.Warm);
    }
</script>
</div>
```

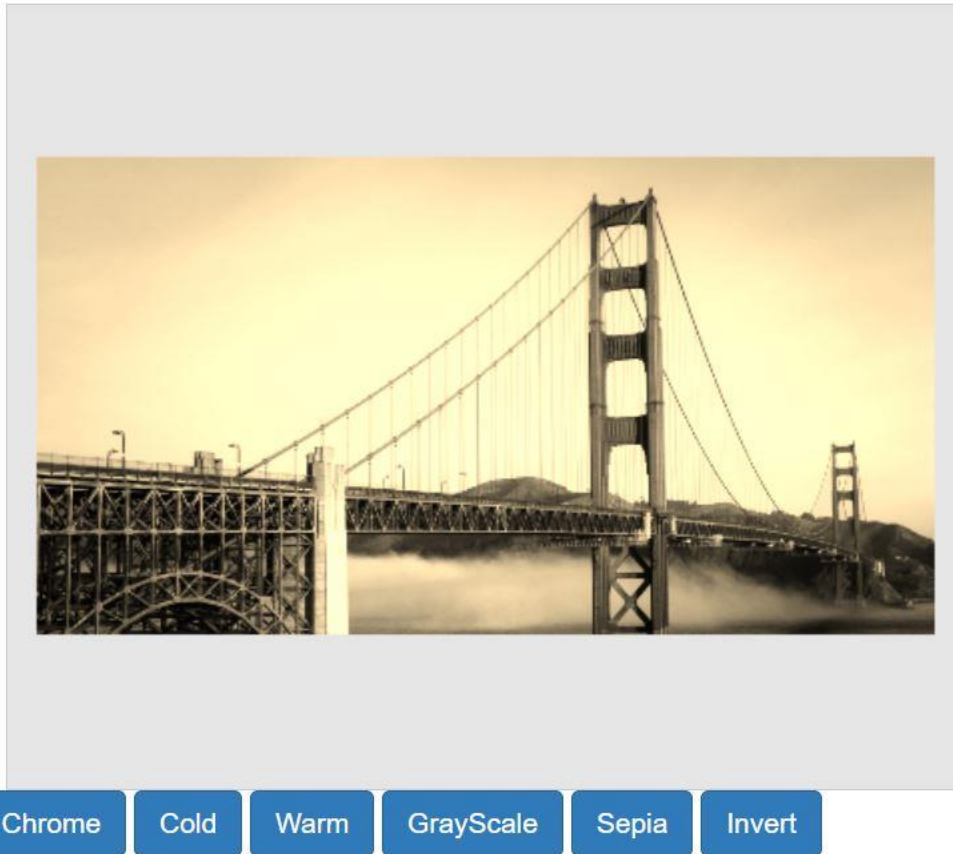


```
    }
    function grayScaleClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.applyImageFilter(ej.imageeditor.ImageFilterOption.Grayscale);
    }
    function sepiaClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.applyImageFilter(ej.imageeditor.ImageFilterOption.Sepia);
    }
    function invertClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.applyImageFilter(ej.imageeditor.ImageFilterOption.Invert);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>
```

DEFAULT.CS

```
public ActionResult Default()
{
    return View();
}
```

Output be like the below.



[Image filtering event](#)

The `[imageFiltering]` event is triggered when applying filtering on the image. This event is passed an object that contains information about the filtering event, such as the type of filtering.

The parameter available in the `ImageFilterEventArgs` event is,

`ImageFilterEventArgs.filter` - The type of filtering as `ImageFilterOption` to be applied in the image editor.

`ImageFilterEventArgs.cancel` – Specifies to cancel the filtering action.

[Finetune in the ASP.NET MVC Image Editor control](#)

Fine-tuning involves making precise adjustments to the settings of an image filter in order to achieve a specific desired effect. It provides control over the intensity and specific aspects of the filter's impact on the image. For example, fine-tuning allows you to modify parameters like brightness, saturation, or other relevant properties to fine-tune the level or quality of the filter's effect. This level of control enables you to achieve the exact look or outcome you want for your image.

[Adjust the brightness, contrast, or sharpness](#)

The `finetuneImage` method is designed to facilitate fine-tuning operations on an image. It accepts two parameters: the first parameter is `ImageFinetuneOption` which determines the type of fine-tuning to be applied (brightness, contrast, or sharpness), and the second parameter represents the fine-tuning value, indicating the degree or intensity of the adjustment. This method allows for convenient adjustment of brightness, contrast, or sharpness by specifying the desired type and corresponding value.

The `finetuneImage` method is used to perform brightness, contrast, or sharpness fine-tuning by specifying this type as a first parameter and specifying the fine-tuning value as the second parameter of the method.

Here is an example of brightness, contrast, and sharpness fine-tuning using the `finetuneImage` method.

CSHTML

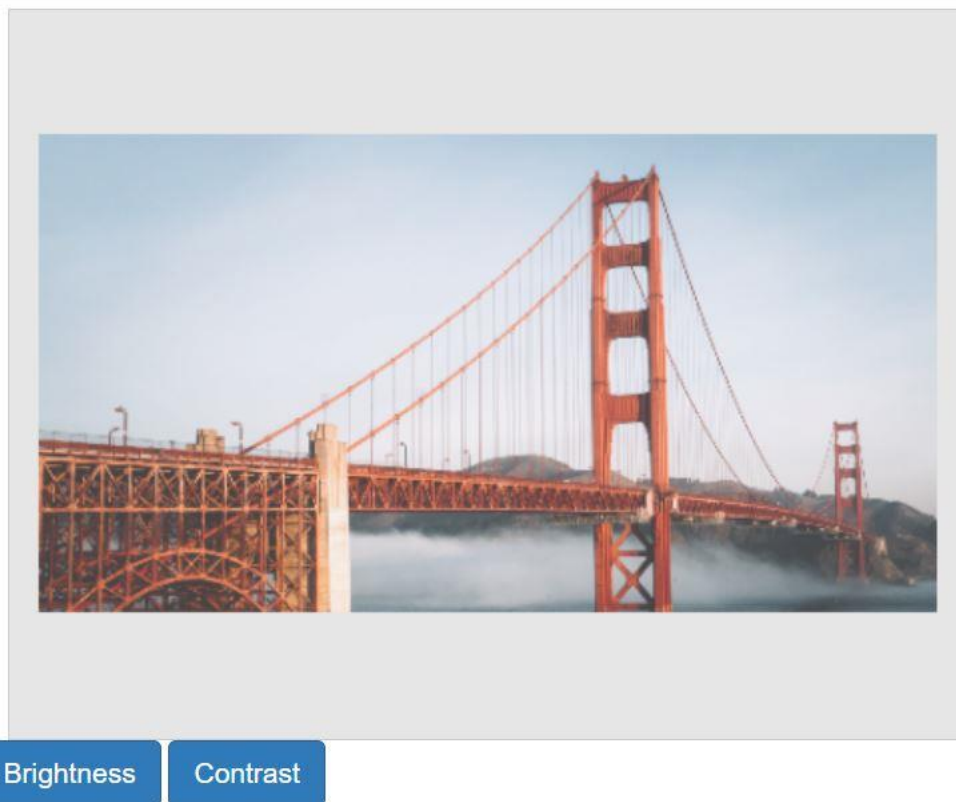
```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").ToolBar(new
string[] { }).Render()
</div>
@Html.EJS().Button("brightnessClick").CssClass("e-
primary").Content("Brightness").Click("brightnessClick").Render()
@Html.EJS().Button("contrastClick").CssClass("e-
primary").Content("Contrast").Click("contrastClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function brightnessClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.finetuneImage(ej.imageeditor.ImageFinetuneOption.Brightness,
10)
    }
    function contrastClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.finetuneImage(ej.imageeditor.ImageFinetuneOption.Contrast,
100);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
```

```
        height: 100%;  
    }  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Adjust the hue, exposure, blur, or opacity

The `finetuneImage` method is designed to facilitate fine-tuning operations on an image. It accepts two parameters: the first parameter is `ImageFinetuneOption` which determines the type of fine-tuning to be applied (hue, exposure, or blur), and the second parameter represents the fine-tuning value, indicating the degree or intensity of the adjustment. This method allows for convenient adjustment of hue, exposure, or blur by specifying the desired type and corresponding value.

Here is an example of hue, exposure, and blur fine-tuning using the `finetuneImage` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">  
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new  
    string[] { }).Render()
```

```

</div>
@Html.EJS().Button("blurClick").CssClass("e-
primary").Content("Blur").Click("blurClick").Render()
@Html.EJS().Button("exposureClick").CssClass("e-
primary").Content("Exposure").Click("exposureClick").Render()
@Html.EJS().Button("hueClick").CssClass("e-
primary").Content("Hue").Click("hueClick").Render()
@Html.EJS().Button("opacityClick").CssClass("e-
primary").Content("Opacity").Click("opacityClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function blurClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.finetuneImage(ej.imageeditor.ImageFinetuneOption.Blur, 10)
    }
    function exposureClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.finetuneImage(ej.imageeditor.ImageFinetuneOption.Exposure,
10);
    }
    function hueClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.finetuneImage(ej.imageeditor.ImageFinetuneOption.Hue, 10)
    }
    function opacityClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.finetuneImage(ej.imageeditor.ImageFinetuneOption.Opacity,
10);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }

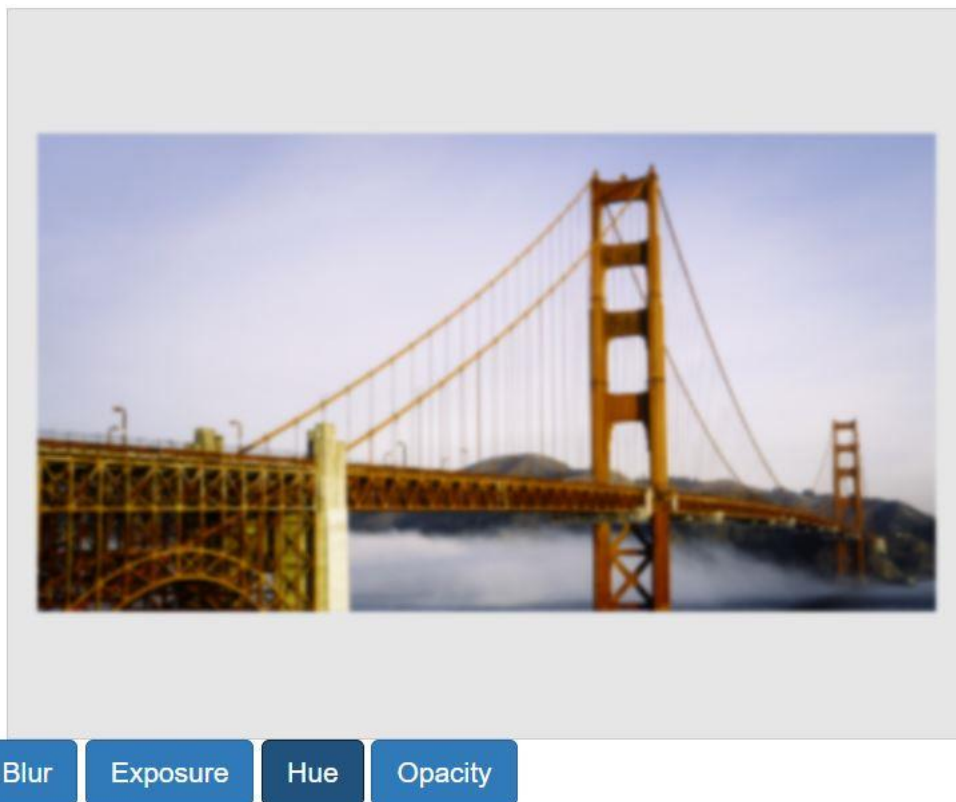
```

```
    }  
    }  
    .control-wrapper {  
        height: 100%;  
    }  
</style>
```

DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.



Finetune value changing event

The `finetuneValueChanging` event is triggered when performing fine-tuning on the image. This event is passed an object that contains information about the fine-tuning event, such as the type of fine-tuning and the value of fine-tuning performed.

The parameter available in the `FinetuneEventArgs` event is,

`FinetuneEventArgs.finetune` - The type of fine-tuning as `ImageFinetuneOption` to be applied in the image editor.

`FinetuneEventArgs.value` - The fine-tuning value to be applied in the image editor.

FinetuneEventArgs.cancel – Specifies a boolean value to cancel the fine-tuning action.

Frames

The frame feature in an Image Editor provides users with the capability to add decorative borders or frames around their images. Frames are a visual design element that can enhance the overall appearance and appeal of an image.

Apply frame to the Image

The `drawFrame` method is a function designed to enable the application of various frame options to an image. This method simplifies the process of adding decorative frames, such as mat, bevel, line, hook, and inset, to an image by allowing users to specify their desired frame type.

Depending on the frame type selected, users may have additional customization options, such as adjusting the frame's thickness, color, texture, or other attributes. This allows for fine-tuning the appearance of the frame to match the image's theme or the user's preferences

The `drawFrame` method in the Image Editor control takes nine parameters to define the properties of the frame to the image:

- `frameType` - Specified the image data or url of the image to be inserted.
- `Color` - Specifies the color for the frame.
- `gradientColor` - Specifies the gradient color for the frame.
- `size` - Specifies the size of the frame.
- `inset` - Specifies the inset value for line, hook, and inset type frames.
- `offset` - Specifies the offset value for line and inset type frames.
- `borderRadius` - Specifies the border radius for line type frame.
- `frameLineStyle` - Specifies the frame line style for line type frame.
- `lineCount` - Specifies the line count for the line type frame.

Here is an example of Frame using the `drawFrame` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("matbtnClick").CssClass("e-
primary").Content("Chrome").Click("matClick").Render()
@Html.EJS().Button("bevelbtnClick").CssClass("e-
primary").Content("Cold").Click("bevelClick").Render()
@Html.EJS().Button("linebtnClick").CssClass("e-
primary").Content("Warm").Click("lineClick").Render()
@Html.EJS().Button("insetbtnClick").CssClass("e-
primary").Content("GrayScale").Click("insetClick").Render()
@Html.EJS().Button("hookbtnClick").CssClass("e-
primary").Content("Hook").Click("hookClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
```

```

        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
    } else {
        imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
    }
}
function matClick() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    imgObj.drawFrame(ej.imageeditor.FrameType.Mat, 'red', 'blue', 20,
20, 20, 20, ej.imageeditor.FrameLineStyle.Solid, 1);
}
function bevelClick() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    imgObj.drawFrame(ej.imageeditor.FrameType.Bevel, 'red', 'blue', 20,
20, 20, 20, ej.imageeditor.FrameLineStyle.Solid, 1);
}
function lineClick() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    imgObj.drawFrame(ej.imageeditor.FrameType.Line, 'red', 'blue', 20,
20, 20, 20, ej.imageeditor.FrameLineStyle.Solid, 1);
}
function insetClick() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    imgObj.drawFrame(ej.imageeditor.FrameType.Inset, 'red', 'blue', 20,
20, 20, 20, ej.imageeditor.FrameLineStyle.Solid, 1);
}
function hookClick() {
    var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
    imgObj.drawFrame(ej.imageeditor.FrameType.Hook, 'red', 'blue', 20,
20, 20, 20, ej.imageeditor.FrameLineStyle.Solid, 1);
}
</script>
<style>
    .image-editor {
        margin: 0 auto;
    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```


DEFAULT.CS

```
public ActionResult Default()  
{  
    return View();  
}
```

Output be like the below.

**Frame changing event**

The **frameChanging** event is triggered when applying frame on the image. This event provides information encapsulated within an object, which includes details about the frame applied in an image. This information encompasses:

Frame Type: This indicates the specific type of frame being applied, whether it's a mat, bevel, line, or hook.

Customization Values: These values contain information about any adjustments or modifications made to the frame. For instance, if the frame can be customized with attributes like color, size, or style, these details are conveyed within the event object.

The parameter available in the **FrameChangeEventArgs** is

- **FrameChangeEventArgs.previousFrameSetting** - The frame settings including size, color, inset, offset, gradient color which is applied before changing the frame.
- **FrameChangeEventArgs.currentFrameSetting** - The frame settings including size, color, inset, offset, gradient color which is going to apply after changing the frame.
- **FrameChangeEventArgs.cancel** - Specifies a boolean value to cancel the frame changing action.

Resize

The resize feature in an Image Editor is a valuable tool that empowers users to modify the size or dimensions of an image to meet their specific requirements. Whether it's for printing, web display, or any other purpose, this feature allows users to tailor images to their desired specifications.

Apply resize to the image

The Image Editor control includes a `resize` method, which allows you to adjust the size of an image. This method takes three parameters that define how the resizing should be carried out:

- **width:** Specifies the resizing width of the image.
- **height:** Specifies the resizing height of the image.
- **isAspectRatio:** Specifies a boolean value indicating whether the image should maintain its original aspect ratio during resizing. When set to true, the image will be resized while preserving its aspect ratio

Here is an example of resizing the image using the `resize` method.

CSHTML

```
<div class="col-lg-12 control-section e-img-editor-sample">
    @Html.EJS().ImageEditor("image-editor").Created("created").Toolbar(new
string[] { }).Render()
</div>
@Html.EJS().Button("aspectbtnClick").CssClass("e-primary").Content("Aspect
ratio").Click("aspectClick").Render()
@Html.EJS().Button("nonaspectbtnClick").CssClass("e-primary").Content("Non
Aspect Ratio").Click("nonaspectClick").Render()
<script>
    function created() {
        var imageEditorObj =
ej.base.getComponent(document.getElementById('image-editor'), 'image-
editor');
        if (ej.base.Browser.isDevice) {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/flower.png');
        } else {
            imageEditorObj.open('https://ej2.syncfusion.com/demos/src/image-
editor/images/bridge.png');
        }
    }
    function aspectClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.resize(300, 400, true);
    }
    function nonaspectClick() {
        var imgObj = ej.base.getComponent(document.getElementById('image-
editor'), 'image-editor');
        imgObj.resize(300, 400, false);
    }
</script>
<style>
    .image-editor {
        margin: 0 auto;
```

```

    }
    .e-img-editor-sample {
        height: 80vh;
        width: 100%;
    }
    @@media only screen and (max-width: 700px) {
        .e-img-editor-sample {
            height: 75vh;
            width: 100%;
        }
    }
    .control-wrapper {
        height: 100%;
    }
</style>

```

DEFAULT.CS

```

public ActionResult Default()
{
    return View();
}

```

Output be like the below.

![ImageEditor Sample](images/image-editor-resize.png)

Resizing event

The **resizing** event is triggered when resizing the image. This event provides information encapsulated within an object, which includes details about the previous and current height and width of an image.

The parameter available in **ResizeEventArgs** is,

- **ResizeEventArgs.previousWidth** - The width of the image before resizing is performed.
- **ResizeEventArgs.previousHeight** - The height of the image before resizing is performed.
- **ResizeEventArgs.width** - The width of the image after resizing is performed.
- **ResizeEventArgs.height** - The width of the image after resizing is performed.
- **ResizeEventArgs.isAspectRatio** - The type of resizing performed such as aspect ratio or non-aspect ratio.
- **ResizeEventArgs.cancel** - Specifies a boolean value to cancel the resizing action.

Accessibility in Image editor control

The Image Editor component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Image Editor component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

The Image Editor component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Image Editor component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + Z** | **Undo the last user action.** |

| **Ctrl + Y** | **Redo the last user action.** |

| **Ctrl + S** | **To save the Image.** |

| Ctrl + O | To open the Image. |

| Delete | To delete the shape once the shape got selected through mouse click . |

Ensuring accessibility

The Image Editor component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Image Editor component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Image Editor component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)